Personal Computer

NEC PC-8801 +mkII

マシン語活用入門

塚本浩二＝著

ナツメ社

[本書を読まれる方へ]

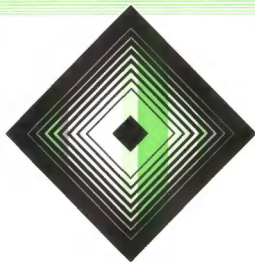
⊙マイコン・ユーザーの中には、
ぜひマシン語をマスターしたい!
と考えている人が、いまだに多いようです。
BASICでは遅すぎてビジネスに活用できない、
マシン語でゲームを作ってみたい、
より深くマイコンのことを知りたい、
マシン語がマスターできるとカッコイイ!
マシン語こそ「言語の中の言語」だから、
と、その理由もまちまちです。
⊙そういった要望にこたえてか、
マシン語に関する本がたくさん出版されています。
しかし、残念ながら、
マシン語をマスターできる人は
数千人に1人の割合といわれています。
⊙では、マシン語はほんとうに
むずかしいのでしょうか? 答えはノーです。
ただ、BASICより
手間がかかることは確かです。
「手間がかかる」と「むずかしい」は別のもの、
根気よくやれば必ずマスターできます。
⊙この本が、そういった、
マシン語マスターへの熱意をもった
あなたのお手伝いをします。
見やすくわかりやすい2色刷の編集ですし、
そのうえ、画面表示も
できるだけ多く採用しました。
さあ、この本でマシン語にチャレンジ!

cover design——nobuo nakagaki＋yoshifumi hayase

Personal Computer

NEC PC-8801 +mkII

マシン語活用入門

塚本浩二＝著

ナツメ社

まえがき

　現在、

　　　　　ＰＣ-8801・ＰＣ-8801mkⅡ

は

　　　　　最高の８ビット・マシン

といわれています。

　しかし

　　　　　ＢＡＳＩＣでは、物足りない

とお感じではないでしょうか？

　実は、ＰＣ-8801やＰＣ-8801mkⅡの性能が高すぎるため、ＢＡＳＩＣでは力を十分に発揮できないのです。

　どうしたらよいのでしょうか？

　この問題を解決するには

　　　　　マシン語を利用すればよい

のです。

　が、一般的には

　　　　　マシン語を学ぶのは難しい

といわれています。

　あなたは、どうお思いでしょうか。

　もし、あなたが

　　　　　マシン語は難しい

そう思ってしまっているのでしたら、本書を御一読することをおすすめいたします。

　本書では、まず

　　　　　なぜ、マシン語が難しいと思われてしまっているのか

を考え、次に

　　　　　マシン語入門

そして、マシン語を活用するうえで重要でありながら、今まで一部のシステム・エンジニア等にしか知られていなかった

　　　　　内部ルーチン

へと進んでいきます。

　本書を読み終った時、あなたは

　　　　　マシン語は難しくない

そう断言できることでしょう。

フォーサイト企画部　塚 本 浩 二

もくじ

第1章　マシン語への招待――7

1・1　マシン語とは何か――8
1・2　なぜ、マシン語なのか――12

第2章　マシン語を活用する前に――19

2・1　マシン語は難しくない――20
2・2　ニーモニックは人間的だ――22
2・3　マシン語マスターへの近道――27
2・4　内部ルーチンを使え！――30

第3章　マシン語の基礎――39

3・1　この章で学ぶこと――40
3・2　マシン語の基礎知識――41
3・3　レジスタとは――49
3・4　フラグとは何か――52
3・5　ＬＤ命令――55
3・6　演算命令――59
3・7　ＪＰ命令とＣＰ命令――64
3・8　ＣＡＬＬ命令とＲＥＴ命令――70

第4章　内部サブルーチンの活用──77

4・1　本章を読む前に──78

4・2　アセンブラのための命令──79

4・3　内部ルーチンの活用！──85

4・4　マシン語からBASICへ──92

4・5　マシン語からモニタへ──97

4・6　WIDTHをマシン語で──101

4・7　画面設定内部サブルーチンあれこれ──113

4・8　PRINTするにはどうするか──123

4・9　１文字表示内部ルーチン──128

4・10　マシン語で文字列をPRINT！──149

第5章　次のステップへ──157

5・0　おわりに──158

第6章　μCOM-82インストラクション活用表──161

第1章

マシン語への招待

◆

1·1 マシン語とは何か

皆さんは、マイコン・ショップなるものをご存じですか?

マァ、皆さんの中に

マイコン・ショップなんて知らないぞ

なんて人は、いるはずがないとは思いますが、万が一、ということで説明させてもらうことにいたしましょう。

マイコン・ショップは、マイコンに関するもの

パソコン

周辺機器(ディスプレイやプリンタのことデス)

ソフトウェア

マイコンに関する書籍

はたまた

パソコンの修理

マイコンに関する質問・相談

にまで応じてくれるという、我々マイコン・ファンには、寺のお坊さんの説教よりも、ずっとありがたーい店のことなのです。

いわば

マイコンよろず屋

とでもいうようなものなんですね、ハイ。

【マイコン界はマシン語だらけ】

ワタクシも、ちょくちょくマイコン・ショップには行きます。

別に――特に何か用がある――わけじゃあないんですね。とにかくたまに秋葉原のマイコン・ショップに行ってウロウロしないと、落ちついて仕事ができないというか、なんとなくイライラするというか、そういう状態になってしまうわけです。

そういうわけで、マイコン・ショップに行ってみると、

ゲームのカセット

が、ずらーっと並んでいるんですね。

おもしろそうなゲームをちょっと並べてみましょう。

インベーダ　(マシン語+BASIC)

ラリー・X　(オールマシン語)

スーパー・ムービング・ブロック　(オールマシン語)

平安京エイリアン　(BASIC+マシン語)

ギャラクシアン　(オールマシン語)

パックマン　(オールマシン語)

気がつかれた方も多いと思いますが、すべてに

マシン語

と表示されていますね。私の見たところ

カラー・グラフィックで

スピーディーなゲーム

には、ほとんど、このマシン語なる単語が表示されているようでした。なぜでしょうか。

ゲーム・カセットの方はこのくらいにして、今度は書籍の方を見てみることにしましょう。

ＰＣ8801マシン語入門

マシン語活用ハンドブック

やさしいマシン語入門

マシン語ゲームの作り方

フムフム、やはり書籍の方も――マシン語――が幅をきかせているようです。

ここでもマシン語、あっちでもマシン語、右を見ても左を見ても、マシン語、マシン語、マシン語――。

いったい

マシン語って何なんだ!

【これがマシン語だ】

マァ、叫んだところでわかるわけはありませんね。とにかく

マシン語

なるものを見てみようじゃないですか。

まず、ＰＣ-8801のリセット・ボタンを押してください。押しましたか？　画面には

```
Disk version (Aug 20,1982)
How many files(0-15)? 3
NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
45944 Bytes free
Ok
■
```

と表示されましたね。

おっと

おかしい、違ってるぞ

という方が何人かいらっしゃるようです。

たぶん

NEC N−88 BASIC Ver 1.1

の所が違っているのでないでしょうか？

皆さんは、驚かれるかもしれませんが、じつは

のです。

NEC N-88 BASIC Version 1.0	旧 PC−8801
NEC N-88 BASIC Version 1.1	新 PC−8801
NEC N-88 BASIC Version 1.3	PC−8801mkII

おわかりいただけたでしょうか。さて、ここで

```
mon
```

とキーインしてみてください。すると、画面には

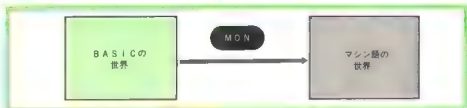
```
Disk version [Aug 20 1982]
How many files(0-15)? 0
NEC N-88 BASIC Version 1.1
OK
mon

n]
```

〈h］〉が表示されるわけです。

『ん？ それがどうしたんだ』

と思われる方も多いかもしれませんね。しかし、いつの間にか、皆さんは

BASICの世界	MON →	マシン語の世界

ここまで来れば、あとはカンタンです。

```
d0000,d002f
```

とキーインすると

何やら、わけのわからない数字のようなものがあらわれました。

〈注〉上の画面は40桁モードです。

BASICのデータ？

インベーダの行列？

否。このわけのわからない数字こそ、我々が探し求めていた

なのです！

ちょっと、BASICのプログラム・リストと比較してみましょう。

```
10 '*******************************
20 '*     デモンストレーション プログラム       *
30 '*                for PC-8801    *
40 '*                   by K.ツカモト  *
50 '*******************************
60 '
100   FOR I=1 TO 100
110      PRINT "K.ツカモト  ";
120   NEXT
130 '
140 END
```

↑BASIC

1·2　なぜ、マシン語なのか

いやはや、マイコン界では

マシン語、マシン語

というふうに——マシン語だらけ！　　のようです。

なぜ

犬も歩けば

マシン語にあたる

ほどに、マシン語が使われ、また、マシン語を学びたいという人がいるのでしょうか。どう、ひいき目で見ても

```
0000 F3 31 FF FF C3 3B 00 00 C3 6A 00 C3 57 17 AB F0
0010 C3 59 42 C3 6A 00 DA 0C C3 A6 40 F3 0B C3 7E 50
0020 C3 DA F1 C3 9C 27 [illegible] 0C C3 DD F1 C3 60 0D 46 0C
```

というプログラムでは、BASICと比較して作りやすいとは思えませんね。実際、かなり長い期間マシン語とつき合ってきた私でさえも、BASICの方が扱いやすいのですから——。

こんな

扱いにくいマシン語

をどうして使いたがるんでしょうかね。何か——特別な利点——があるに違いありません。

その特別の利点とは

スピードがBASICの

数百倍から数千倍である

ということなのです。

マァ、他にもいくつか利点はありますが、この——スピードがBASICの数百倍から数千倍である——という利点の前では、ほとんど無いも同じなのです。

[マシン語はスピード]

しかし、いくら

マシン語は速い

といわれても、どのくらい速いのか皆目見当がつきません。

うーむ、どのくらい速いのだろうか？

と悩んでみても、何もわかりませんね。

とにかく

実験

してみることにいたしましょう。実験こそが、マシン語・マスターへの近道なのです。

マシン語のスピードを実感するには

BASICとマシン語で
同じ内容のプログラムを作って
スピードを比較してみる

のが一番でしょう。

そこで

画面じゅうを
〈●〉で埋めつくすプログラム

を、BASICとマシン語で組んでみて、そのスピードの差を比較してみることにしました。

これならば

マシン語のスピード
を実感できる

のではないでしょうか？

[トロトロのろいBASIC]

まず、次のプログラムをキーインしてみてください。

```
10 '******************************
20 '*  CRT ヲ '●' デ ウメル プログラム  *
30 '*              for PC-8801     *
40 '*                  by K.ツカモト *
50 '******************************
60 '
100 WIDTH 80,25 :CONSOLE 0,25,0,1
110   COLOR 7,0 :PRINT CHR$(12);
```

初期設定

```
120 '
130   FOR Y=0 TO 24
140     FOR X=0 TO 78
150       LOCATE X,Y
160         PRINT "●";
170     NEXT
180   NEXT
190 '
200 END
```

さて、ちゃんとキーインできましたか？

キーインできたら、今度は

```
RUN
```

とキーインしてください。すると、画面には

というように〈●〉が左上から順番に表示されていきます。

ところが、ところが

遅いんですよ、コレが。

まあ、トロトロ、トロトロとじつにゆっくり〈■〉をひとつずつ表示していくわけです。

気のみじかい方は、イライラして

途中で（STOP）キーを

押してしまった！

のではないでしょうか。

そのくらい　遅い！——のです。

【驚異のマシン語】

さて、今度は

マシン語

の方を実験してみましょう。はたして、どのくらいのスピードなんでしょうか、楽しみですね。

初めに、マシン語のプログラムをお見せしましょう。

```
B900 21 C8 F3 11 28 00 0E 19
B908 06 50 36 EC 23 10 FB 19
B910 0D 20 F5 3E 04 D3 31 CD
B918 47 60
```

ヤヤッ！

コレで、ちゃんと動くのか？

と疑いたくなるような——短いプログラム——ですね。あまりの短さに、このプログラムを作った私自身

本当に、動くだろうか

と不安になってしまっているのです。

とにかく、動くか動かないかは、プログラムを走らせてみればわかるのです。

さあ、プログラムを走らせ——おっと！　その前に、このマシン語のプログラムをキーインするのを忘れていました。

まず、

```
WIDTH 80,25
CONSOLE 0,25,0,0
```

とキーインして、画面のモードを

８０字×２５行
白黒モード
ファンクション・キー表示なし

にします。次に下記のようにキーインします。

```
mon
sb900
```

このような画面になりましたか？

このようになっていれば、あとは

です。先ほどのマシン語のプログラム・リストを見ながら

```
21 c8 f3 11 28 00 0e 19
06 50 36 ec 23 10 fb 19
0d 20 f5 3e 04 d3 31 cd
47 60
```

とキーインしていけばいいのです。この時，画面は

となっているはずですが，どうですか？

さて，これで

マシン語プログラムを

キーインし終った

わけです。

キーインし終ったのですから

プログラムを走らせよう！

と言いたいところですが、一応、確かにキーインできたか確認しておきましょう。はっきり言って、私も

一刻も早く

プログラムを走らせたい

のです。

がしかし、マシン語の場合、プログラムに１つでもミスがあると、

タイヘンなことが起こってしまう

のです。いわゆる

暴走

という、世にも恐ろしい、泣く子もだまる、聞くも涙語るも涙の状態になってしまうわけです。コレには私も泣かされました――。

さて、確認してみましょう。

```
db900,b919
```

とキーインすると、先ほどキーインしたマシン語プログラムが

表示されるわけです。ミスはないようです。

さあさあ、今度こそ

プログラ[illegible]せ[illegible]しょう！

マシン語は、恐ろしくスピードが速いですから、〈[illegible]〉で画面じゅうを埋める（８０×２５＝２０００コも[illegible]を表示するのデス）というタイヘンな作業も、ほんの一瞬で終ってしまいます。

ですから

よく目をあけて

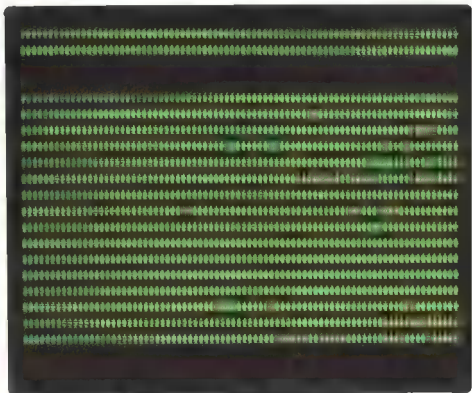
見てください

それでは

```
gb900
```

とキーインしてください。

どうです？

あっ！　という[illegible]に

となってしまいました。

これが

なのです。BASICで、トロトロ〈[illegible]〉をひとつずつ表示していたのとは、えらい違いです。

いやあ

驚きました!

たぶん、皆さんも驚かれたのではないでしょうか? マシン語はBASICの数百倍—— といっても

ここまで速いとは

思わなかった

のではないでしょうか。

最後に、もう一度言わせてもらいたいと思います、

これが、これこそが

なのです!!

2

マシン語を活用する前に

2・1　マシン語は難しくない

マイコン・ユーザーの中の多くの方は

ぜひ、マシン語を

マスターしたい！

と思われているようですね。

その理由も

- **BASICでは遅すぎてビジネスに活用できない**
- **マシン語でゲームを作ってみたい**
- **より深くマイコンのことを知りたい**
- **マシン語がマスターできるとカッコイイ！**
- **マシン語こそ「言語の中の言語」だから**

など、多種多様です。

そのユーザーの要求に答えて、マシン語に関する書籍が数多く出版されています。

しかし、数多くの入門書・参考書が出版されていながら、いまだに、マシン語をマスターできる人は

数千人に１人の割合

といわれています。

また、こまったことに

BASICはやさしいが

マシン語は難しい

と、一般に思われているようです。

このように

数千人に１人の割合

マシン語は難しい

などと聞くと、一応なりともマシン語とつきあいのある私は

うーむ、私は、数千人に１人の

天才だったのか、フムフム――。

と喜ぶわけです。

ところが、ところがです。現実に返ってみると

仕事ができる！

わけもなく、学生の頃

成績が良かった！

わけでもないのです。それどころか、両方とも

低空飛行で

どうにか飛んでいる

ようなありさまなのデス、ハイ。

　このようなワタクシでさえ、マシン語をマスターできたのです。

　ここで、私は声を大にして言いたい。

マシン語は難しくないの

と。

　確かに、マシン語はBASICと比較して

スピードが数百倍速い

かわりに

手間が数十倍かかります

　たったひとつのデバッグ（プログラム中の間違いを見つけること）に——2、3週間かかる——のはザラです。

　が、しかし

手間がかかるから、難しい

わけではないのです。

　それでは、なぜ

マシン■は難しい！

と思いこんでしまっている方が多いのでしょうか？

　それがわかれば、マシン語をマスターできるのでは。

　また、マシン語を『活用する』には、まず、その辺を理解することが必要ではないでしょうか。

　この章では

私の知人に質問する

ことによって、なぜ——マシン語は難しい！　と思いこんでしまったかを探っていこうと思っております。

2·2 ニーモニックは人間的だ

A君　高校1年。

彼は、BASICはほぼマスターし、プログラムもすでに4、5本作ってはいますが――どうもマシン語は――と、マシン語を敬遠しているようです。

なぜ、マシン語を敬遠するのか？

その辺を中心に質問することにいたしましょう。

つか：BASICで、どんなプログラム作ってるの？
A君：うーん、やっぱり、ゲームが多いなあ。
つか：ゲーム作ってて BASICじゃ遅くない？
A君：遅い。特にエイリアンとかがいっぱい動かす時は、じれったくなっちゃうなあ。
つか：それだったら「マシン語」を使えば。
A君：マシン語――。
つか：そう。どうしてマシン語を使わないの？
A君：だって、マシン語って、わけがわからないでしょ。
つか：わけがわからないって？
A君：ホラ、マシン語って――0E、4D、C9――って、これじゃ難しすぎるよ。

フーム、A君は、マイコン専門誌等に、マシン語が

```
F3 31 FF FF C3 [illegible] 00 00 C3 6A 00 C3 57 17 AB F0
```

というふうに載っているのを見て

「あー、マシン語って難しい」

と思いこんでしまったようですね。

このA君のように思いこんでしまっていた方も多いのではないでしょうか？

確かに

```
C3 3B 00 00 C3 6A 00 C3 57 17 AB F0
```

では――マシン語は難しい――と思ってしまうのも当然のような気がします。

はっきりいって、これでは私もマシン語をマスターする気にはなりません。

「それじゃあ、どうやってマスターしたんだ？」

実は、マシン語には

C3 3B 00 00 C3 6A 00 C3 57 17 [illegible] F0

というような、わけの分からない『機械的』な表現の他に、もう1つ

ニーモニック

なる表現の方法があるのです。

ニーモニックとは

> **機械的なマシン語を**
> **人間的な表現に変換したもの**

なのです。

マシン語	ニーモニック
3C	INC A

上の命令は、両方とも

Aレジスタの値に
1を加える

という意味を表します。（Aレジスタなる単語は、BASICでいえば『変数』のような物です）

同じ命令であっても

3C

では、よほどの——マシン語マニア——でなければ理解できないでしょうね。

ところが

INC A

となると——INCREMENT（増加）——というように、英単語を省略した形になっていますから

BASICと
さほどかわらず理解しやすい

わけです。

どうです？　これでA君の言う

マシン語は、わけがわからない

というのが——単なる誤解であった——ことが皆さんに十分理解してもらえたのではないでしょうか。

さて、ここで1章で皆さんにキーインしていただいた

画面をすべて〈●〉

で埋めつくすマシン語プログラム

```
B900 21 C8 F3 11 28 00 0E 19
B908 06 50 36 EC 23 10 FB 19
B910 0D 20 F5 3E 04 D3 31 CD
B918 47 60
```

を

分かりやすいニーモニック

で表してみましょう。

```
                      ; ******************************
                      ; *   CRT ヲ ´●´ デ゛ ウメル フ゜ログ゛ラム   *
                      ; *          for PC-8801         *
                      ; *                  by K.ツカモト *
                      ; ******************************
                      ;
F3C8                  VRAMAD:EQU  0F3C8H
0028                  ADD40: EQU  40
                      ;
                             ORG  0B900H  ← プログラムを[illegible]Hから作る
アドレス マシン語        ;
B900  21C8F3                 LD   HL,VRAMAD  ← HLレジスタにF3C8Hを代入
B903  112800                 LD   DE,ADD40   ← DE[illegible]スタに40[illegible]
                      ;
B906  0E19                   LD   C,25
B908  0650            LOOP1: LD   B,80
B90A  36EC            LOOP2: LD   (HL),´●´  ← HLの値のVRAMに[illegible]
B90C  23                     INC  HL
B90D  10FB                   DJNZ LOOP2
B90F  19                     ADD  HL,DE
B910  0D                     DEC  C
B911  20F5                   JR   NZ,LOOP1
                      ;
B913  3E04                   LD   A,4
B915  D331                   OUT  (31H),A
B917  CD4760                 CALL 6047H
```

どうです？

BASICと

たいして違わない

のではないでしょうか。

これなら

マシン語が分かる

ような気がしてくる

ですね。

その気持ちが

あなたがマシン語へ

一歩ちかづいた証拠

なのです。

[[illegible]のアセンブラ]

ところが！　せっかく

分かりやすいニーモニック

でプログラムを作っても、マシン語に変換できなければしかたないですね。

この

ニーモニック

→マシン語

のための表を

インストラクション活用表

と呼び、本書巻末に掲載してあるのがそれです。

この表を見ながら、

『えーと、ADD　HL，DEは

うーん、あっあった19か』

とやっていけば、マア、マシン語に変換できます。

しかし、これでは、インベーダを作り終えた頃には、曽おじいさんになってしまいます。（ちょっとおおげさでしたね）

この、ニーモニック→マシン語という、めんどうな作業を全て自動的に行うプログラムがあるのです。

一般的に

アセンブラ

と呼ばれているものがそれです。

このアセンブラには

アセンブリ[illegible]

アセンブリ・ランゲージ

はたまた

アッセンブラ

などと呼ぶ方もおられるようですが、どれも同じようなものです。

アセンブラは、単に——ニーモニック➜マシン語——と変換するだけでなく数多くの便利な機能が内蔵されています。

それほど高価なものではありません。

ぜひ、ご自分にあった評判の良いアセンブラを選び使ってみてください。

そして、本書にあるようなニーモニックで表されている短いマシン語プログラムをキーインしてみてください。

いえいえ、プログラムが分からなくてもいいのです。

とにかく、キーインして

マシン語をあなたの身近なものにしてください

それが、マシン語をマスターする

スタート・ライン

なのですから。

2・3　マシン語マスターへの近道

B氏　37歳　会社員

彼は——マシン語をマスターするゾ！——と、マシン語入門書を片手にかなり熱心にマイコンの前に向かった結果、ほぼ入門書の内容を理解できたのです。

これで、マシン語はマスターした——と思ったのですが、実際にプログラムを組んでみようと思うと、手も足もでない。

マシン語はマスターしたはずだが
実際にプログラムを組むことはできない

なぜ、B氏はマシン語のプログラムを組むことができないのでしょうか？

今回は、その辺を中心に質問していきたいと思います。

> **つか**：Bさん、マシン語をマスターした——ということを、もう少し具体的に説明してもらえないでしょうか？
> **B氏**：いや、マスターしたといえる程のものではないのですが。
> **つか**：命令は全て覚えられましたか？
> **B氏**：自分ではほとんどの命令はわかったつもりですし、簡単なプログラムなら時間はかかりますが作れるようになりました。
> **つか**：そこまでできればマスターしたも同然じゃないですか。
> **B氏**：いやいや、実際に何かプログラムを組もうとすると、もう、どこから手をつけてよいやらわからないのです。

このB氏のような方が——マシン語がマスターできない——という方のほとんどだと私は思うのですが、どうでしょうか？

例えば

LD　A，3CH

を見れば

Aレジスタに3CHを代入してる

とわかるけれども、実際にプログラムを組もうと思っても

どこから手をつけてよいのか分からない

というわけです。

定石がマシン語を左右する

このように、多くの方がマシン語マスターに今一歩という所でストップしてしまっているのでしょうか？

私は、このことを説明する時、よく

マシン語を将棋にたとえて説明

します。

将棋で勝負を左右するのは、いかにコマの動きを理解するかではなく、いかに多くの定石を覚えているかなのです。

マシン語にも同じことが言えます。

マシン語の命令は

LD A, 1
→ Aレジスタに1を代入する
INC A
→ Aレジスタの値に1を加える
ADD A, B
→ Aレジスタに B レジスタの値を加える

というように、非常に単純なものばかりです。

ところが、ところがなのです！

単純な命令しかないため

画面に表示しよう

コサインを求めよう

カラーを指定しよう

と思っても、そのような命令は

マシン語には用意されていない

のです。

それどころか

乗算、除算すら無い

のです！

そのため、乗算を行いたい場合には、まず

乗算を行うプログラム

を自分で組まなければならない

のです。

このため、BASICでは

というように数秒でできてしまうプログラムでも、マシン語では1か月以上もかかってしまうわけです。

「おかしいぞ。その割には、長い

マシン語プログラムが数多くあるぞ！」

ごもっともです。実は、1か月以上かかってしまうはずの乗算プログラムを、なんと数時間で作ってしまう方法があるのです。

今まで、マシン語のプログラムは数多く作られてきました。

これらのマシン語プログラムが作られていく過程で、数多くの

マシン語の定石

が作られていったのです。

これらの定石は、改良に改良を重ねられ、非常に完成されたものとなっています。

ですから、このマシン語の定石を利用することによって

短時間で、しかも完成度の高い

マシン語プログラムを作ることができる

わけです。

さて、この定石は、どこにあるのでしょうか？

マシン語定石集

などという書籍は、どこへ行ってもないのです。

将棋の定石が、実戦から現れてきたように、マシン語の場合も、実戦——即ち

数々のマシン語プログラムの中にある！

のです。

しかし、いくら——マシン語プログラムの中にある——といわれても、11ページのディスプレイ表示にあるようなものでは、わけが分かりません。

やはり、人間的なニーモニックで表わされたマシン語プログラムの方がいいですね。

ところが、最近のマシン語プログラムにはニーモニックで表わされているものがほとんどありません。こまったことです。

『マシン語の定石を知りたい』

このような方には、本書にあるような数々のサンプル・プログラムを

ご自分で解析してみる

ことを、私はお勧めいたします。

命令を理解されたのですから、マシン語をマスターするのに今一歩の所まできているのです。

マシン語の定石を知ることによって、残りの一歩を進んでみてはいかがでしょうか？

2·4 内部ルーチンを使え！

さて、３人目、最後です。

Ｃ氏　27歳　会社員。

彼は、数々の難関を乗りこえ実際にマシン語をマスターしたという貴重な方です。すでにいくつかマシン語プログラムを組んでいます。

マシン語をマスターした彼には

これからマスターする方へのアドバイス

を中心に質問することにしましょう。

つか：マシン語をマスターする時の、一番の難関はＣさんの場合、何だったですか？

Ｃ氏：やはり、一番妨げになったのは「マシン語恐怖症」ですね。これがあるうちは、まずムリじゃあないでしょうか。

つか：Ｃさんは、その「マシン語恐怖症」をどうやってなくしたんですか？

Ｃ氏：うーん、まずアセンブラを買ってきました。それで、とにかく短いニーモニックのマシン語プログラムを探してキーインしたんです。そうすると、何か――自分で作ってる――という感じがしてきまして。

　最初はそうではないんですが、何といっても命令自体は単純ですからね。このくらいならオレにでもできるんじゃあないかって思ったわけです。

つか：で、実際にはどうでしたか？

Ｃ氏：いやあ、もちろん、できませんでしたよ。命令１コ１コしか見てなかったんですね。そこで――どうしてそうなるのか――というアルゴリズムを考えることにしたんです。

　初めはほとんど分からなかったけれど段々分かってきて、もうその頃には自分で作れるようになってました。

つか：マシン語でプログラムを組む時に、重要になってくるのは何ですか？

Ｃ氏：いかに手間をかけずに作るかでしょうね。それには、数多くのプログラムを解析して定石を知り、あと内部ルーチンをいかに活用するかがポイントになってくるでしょうね。

　この内部ルーチンの重要性は、意外と知られていないんで特に強調したいですね。

このような的確なアドバイスは、非常に参考になりますね。
しかし、C氏が最後の方で強調していた

とは、いったい何のことでしょうか？
C氏がこれだけ強調して重要性を訴えていたのですから、マシン語を活用するうえで有効なものであることだけは確かなようです。

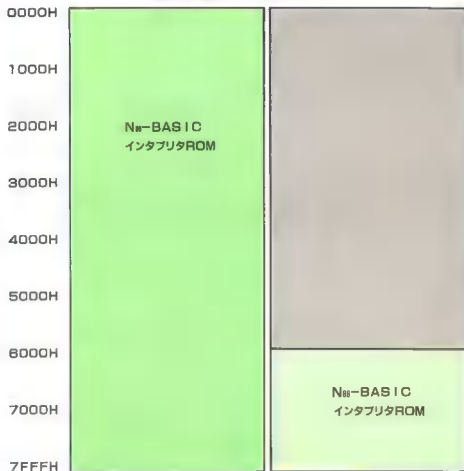
【[illegible]部ルーチンとは】

我々がいつも何の気なしに使っているBASIC。
このBASICは、実は、24Kバイトという長い長～いマシン語のプログラムなのです。

N_{88}-BASIC メモリ・マップ

0000H
1000H
2000H
3000H
4000H
5000H
6000H
7000H
7FFFH

N_{88}-BASIC
インタプリタROM

N_{88}-BASIC
インタプリタROM

ということは、BASICの命令

PRINT
LOCATE
COLOR
BEEP

などと同じ役割をするマシン語プログラムがBASICの中のどこかにあるわけです。

このような

BASICの中のルーチンを

内部ルーチンと呼ぶ

わけです。

【内部ルーチンを使え】

さて、それでは

内部ルーチンを使うと

どれだけトクか

実験してみることにいたしましょう。

これも、やはり

同じ内容のプログラムを組んで

比較してみる

のが一番ではないでしょうか？

今回は

画面クリアを行うプログラム

を

①内部ルーチンを使わないプログラム

②内部ルーチンを使ったプログラム

とで比較してみることにします。

【内部ルーチンを使わないと】

まずは

内部ルーチンを使わないプログラム

の方から実行してみましょう。

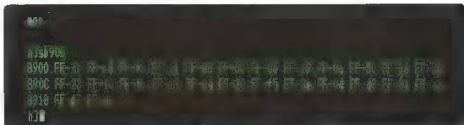
⬇マシン■

```
B900 21 C8 F3 11 28 00 0E 19
B908 06 50 36 20 23 10 FB 19
B910 0D 20 F5 3E 04 D3 31 CD
B918 47 60
```

```
                    ; ******************************
                    ; *  CRT ヲ CLEAR スル ブ゚ログ゙ラム    *
                    ; *          for PC-8801         *
                    ; *                 by K.ツカモト *
                    ; ******************************
                    ;
F3C8                VRAMAD:EQU  0F3C8H
0028                ADD40: EQU  40
                    ;
                           ORG  0B900H
                    ;
B900 21C8F3                LD   HL,VRAMAD
B903 112800                LD   DE,ADD40
                    ;
B906 0E19                  LD   C,25
B908 0650           LOOP1: LD   B,80
B90A 3620           LOOP2: LD   (HL),' '
B90C 23                    INC  HL
B90D 10FB                  DJNZ LOOP2
B90F 19                    ADD  HL,DE
B910 0D                    DEC  C
B911 20F5                  JR   NZ,LOOP1
                    ;
B913 3E04                  LD   A,4
B915 D331                  OUT  (31H),A
B917 CD4760                CALL 6047H
```

前に説明したように

```
WIDTH 80,25 :CONSOLE 0,25,0,0
```

とキーインして、画面のモード設定を行ってから

というようにマシン語プログラムをキーインしてください。

キーインし終わりましたら

```
GB900
```

と実行します。すると，画面は

というように

画面クリア

されます。

【内部ルーチンを使えば】

さて、今度は

内部ルーチンを使ったプログラム

ですね。

⬇マシン語

まずは、とにかく実行してみることにしましょう。

```
B900 3E 0C CD 0D 3E 3E 04 D3 31 CD 47 60
```

⬇ニーモニック

```
                 ; ******************************
                 ; *  CRT ヲ CLEAR スル プ゜ログ゛ラム    *
                 ; *            for PC-8801           *
                 ; *                    by K.ツカモト *
                 ; ******************************
                 ;
                        ORG   0B900H
                 ;
B900 3E0C               LD    A,12
B902 CD0D3E             CALL  3E0DH
                 ;
B905 3E04               LD    A,4
B907 D331               OUT   (31H),A
B909 CD4760             CALL  6047H
```

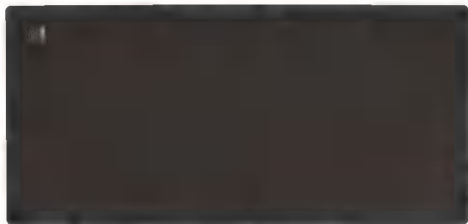
先ほどと同じように

```
WIDTH 80,25 :CONSOLE 0,25,0,0
```

とキーインして画面のモード設定を行ってから、マシン語プログラムを入力してください。

入力し終わりましたら

```
gb900
```

とキーインして、マシン語プログラムを実行します。

どうです？　ちゃんと、画面クリアされましたね。

[内部ルーチンの長所・短所]

それでは

内部ルーチンを使わないプログラム

内部ルーチンを使ったプログラム

を比較してみてください。

見てお分りの通り、内部ルーチンを使うと

近くまで短いプログラムになっていますね。

こんな

画面をクリアする

というカンタンなプログラムでさえ、こんなに差があるのですから

πを求める

画面のモード設定を行う

などというような複雑なプログラムでは、内部ルーチンを使う場合、非常に効果があるのです。

ところが、この便利な内部ルーチンには欠点があるのです。

内部ルーチンを使うと

プログラムの実行速度がやや遅くなってしまう

のです。

まずは、このマシン語のプログラムを見てください。

```
3E0D C5 D5 E5 F5 CD 8B 42 47 3A B6 E6 B7 28 08 EE 19
3E10 32 B6 E6 78 18 05 78 FE 20 38 27 CD 20 62 CD 70
3E2D 44 F1 F5 FE 0A 20 0A 2A 86 EF 26 01 22 86 EF 18
3E3D 11 F1 47 3A B6 E6 FE 19 78 28 03 32 8F EF E1 D1
3E4D C1 C9 FE 19 28 3F FE 1C 30 27 FE 07 20 05 CD 9B
3E5D 3E 18 DE FE 08 28 33 FE 0A 38 D6 20 0E 3A 8F EF
3E6D FE 0D 3E 0A 20 05 11 17 5E 18 11 FE 0E 30 C2 C6
3E7D 0E 21 [illegible] 5D 87 4F 06 00 09 5E 23 56 01 3E 3E C5
3E8D D5 2A 86 EF C9
```

このマシン語プログラムは、いったい何だと思いますか？

注意深い方は、もうお分かりだと思います。

そうです。先ほど使った内部ルーチンの一部なのです。（全部だと、この10倍はあります）

```
;
        ORG     0B900H
;
        LD      A,12
        CALL    3E0DH           内部ルーチンをコール
;
        LD      A,4
        OUT     (31H),A
        CALL    6047H
```

このように内部ルーチンを使うと、組むプログラムは短くてすみますが、実際に動くプログラムは長くなってしまい実行速度が遅くなってしまうわけです。

遅くなってしまう

とはいうもののマシン語ですから、よほどのことがないかぎり不便は感じないでしょうが……、

内部ルーチンは

単に使うのではなく
うまく考えて使う

ことにより、その威力を発揮できるのです。

この

内部ルーチンを活用してこそ
マシン語をマスターしたといえる

のです！

●内部ルーチンの長所・短所

	[illegible]	[illegible]
内部ルーチンを使用した場合	手間がかからない	実行速度がやや遅い
内部ルーチンを使用しない場合	実行速度が速い	手間がかかる

【次のステップへ】

ここまで説明して

『どうやって内部ルーチンを使うんだ？』
『他に便利な内部ルーチンはないのか？』

と思われた方は、非常に多いのではないでしょうか。

本書は、今まで一部のシステム・エンジニアにしかあまり知られていなかった、しかも、もっとも重要な

いかに内部ルーチンを活用するか

を中心に進めてまいります。

しかし、それは、第４章でじっくり説明させてもらうことにして、まずは、第３章へとステップを進めて行くといたしましょう。

第3章

マシン語の基礎

3・1 この章で学ぶこと

前章を読んで、皆さんは、

ぜひマシン語をマスターし

活用してみたい！

と思われたのではないでしょうか？

皆さんにこのように思っていただくと、私もこの本をここまで書いてきた甲斐があるというものです。

フムフム

「いますぐ、内部ルーチンを教えて欲しい」

ごもっともです。

内部ルーチンの重要性を理解されたのですから

内部ルーチンを知りたい！

と思うのも当然のことといえましょう。

しかし、マア、あせらないでください。

この章では

マシン語の基礎

と題して、内部ルーチンを活用するために必要なマシン語の用語・命令を説明していきたいと思っております。

入門

といっても、なにせ1冊の本で入門から活用まで説明しようというのですから、それほど深く説明することはできません。

もっと深い所まで知りたい

という方は、是非、書店で

マシン語入門書

を購入してみてください。

おっと！　すでにご存知の方もおられるようですね。

そのような方は

確認

のつもりで読んでみてください。

それでは、ボチボチ参りましょうか——。

3・2　マシン語の基礎知識

さて、これまでに、いろいろなマシン語に関する専門用語がでてきましたね。

例えば

　　　　アドレス

　　　　ＣＰＵ

　　　　ＲＡＭ

　　　　レジスタ

などです。

前章までは、とにかく

　　　　内部ルーチン活用の[illegible]

を皆さんに理解していただくために、これらの専門用語について説明しませんでした。

まずは、これらの専門用語について説明することにいたしましょう。

【CPUとは】

皆さん、皆さんがお持ちのマイコンのマニュアルを見てください。

ＣＰＵ　　μＰＤ780Ｃ－１（Ｚ80Ａコンパチ）

と載っていますね。

マニュアルにかぎらず、マイコン関係書であればほとんど――ＣＰＵ――という単語がでてきます。

さて、この

　　　　ＣＰＵ

なる単語は、はたしてどのような意味なのでしょうか？

このＣＰＵ、日本語では

　　　　中央処理装置

といい

メモリの中にあるプログラムを解読し そのプログラムを実行する

ものなのです。

いわば

　　　　マイコンの[illegible]

のような存在なわけです。

ＣＰＵが――中央処理装置　　であることはわかりました。さて、

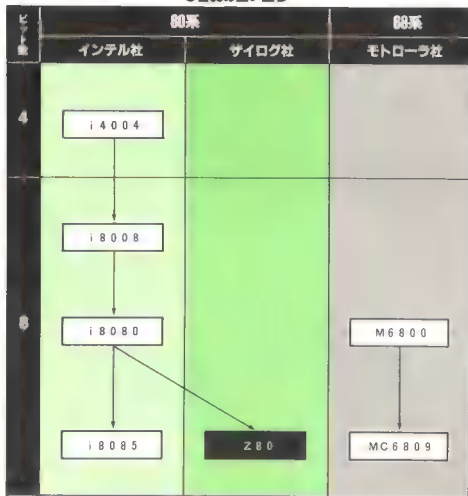
とはいったい何のことでしょうか？　さっぱりわかりませんね。
　実は、コレはＣＰＵの種類を意味するのです。
　ＰＣ－8801の場合、ＣＰＵは

μＰＤ780Ｃ－１

というものを使っているわけです。
　その後に書いてある

（Ｚ80Ａコンパチ）

というのは、この――μＰＤ780Ｃ－１――がザイログ社で設計されたＣＰＵ――Ｚ80Ａ――と同等の機能を持っているということなのです。

●Ｚ80の生い立ち

〔RAMとROM〕

プログラムを実行するのがCPUだというのはわかりました。また、PC－8801はZ80と同等な機能を持ったμPD780C－1というCPUを採用していることもわかってもらえたと思います。

ところで、そのCPUが実行するプログラムは、どこにあるのでしょうか？

実は、この

情報を記憶する装置

のことを

[illegible]

というのです。

このメモリには、大きく分けて

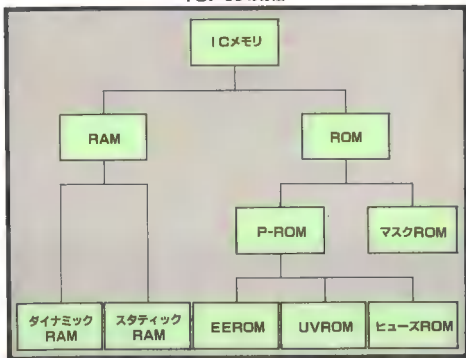
ROM

RAM

という[illegible]つの種類があります。

RAMは

Random Access Memory

ICメモリの分類

の略で

読み書き可能なメモリ

のことをいいます。

我々がマシン語やBASICで組んだプログラムは、このＲＡＭに記憶されているわけです。

ＲＯＭは

Read Only Memory

の略で

読み出し専用のメモリ

のことをいいます。

このＲＯＭは、作る時に書き込んでおけば、後からその内容を変えることができないので、パソコンではBASICを記憶させておくなどに使用しております。

【アドレス】

メモリには、65536個という非常に数多くの記憶場所があります。

ですから、メモリから記憶内容を読みだす場合には

どこの内容を読むのか

また、メモリに書き込む場合

どこに書き込むのか

を指定する必要があるわけです。

このメモリの記憶場所を区別するために、１つ１つに

アドレス

という数字をつけているのです。

このアドレスは

０００００Ｈ～ＦＦＦＦＨ

という４桁の16進数で表わされています。

「もしかしたら、アレがそうなのでは？」

そう。その通りです。

以前でてきたマシン語のプログラム

```
アドレス
0000 F3 31 FF FF C3 3B 00 00
0008 C3 6A 00 C3 57 17 AB F0
0010 C3 59 42 C3 6A 00 DA 0C
0018 C3 A6 40 F3 0B C3 7E 50
0020 C3 DA F1 C3 9C 27 88 0C
0028 C3 DD F1 C3 60 0D 46 0C
```

の中の

```
0000
0008
0010
```

という部分は，マシン語のプログラムが記憶されている場所を指すアドレスだったのです。

メモリとアドレス

0000H	0001H	0002H	0003H	0004H	0005H	0006H	0007H
0008H	0009H	000AH	000BH	000CH	000DH	000EH	000FH
0010H	0011H	0012H	0013H	0014H	0015H	0016H	0017H
0018H	0019H	001AH					
					FFE5H	FFE5H	FFE6H
FFE8H	FFE9H	FFEAH	FFEBH	FFECH	FFEDH	FFEEH	FFEFH
FFF0H	FFF1H	FFF2H	FFF3H	FFF4H	FFF5H	FFF6H	FFF7H
FFF8H	FFF9H	FFFAH	FFFBH	FFFCH	FFFDH	FFFEH	FFFFH

【マシン語は16進数だった】

先ほど、私は皆さんに

アドレスは4桁の16進数だ

と説明しましたね。

さて、この

16進数

とは、どんな数なのでしょうか？

「10進数ならば知っているのだが」

そうです。我々が普通使っている数字は、ほとんど－　10進数－ですね。

10進数というのは、1を加えていって

10になったら

1ケタ上がる

という所からきています。

ということは、16進数は、1を加えていって

16になったら

1ケタ上がる

ような数のことではないでしょうか？

実はその通りなのです。

次の図を見てください。

10進数	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16進数	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

16進数の場合、0から9までは今までの10進数と同じですが、10から15までは、あてはめる数字がないので、

10 → A

11 → B

12 → C

13 → D

14 → E

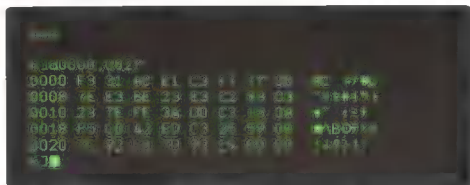
15 → F

というように、アルファベットを使っています。

この16進数を使う場合、10進数等の他の数字と区別するため

末尾にHをつけ区別する

ことになっているのです。

ですから、今まで出てきた

といった、一見わけのわからない

マシン語は　００Ｈ～ＦＦＨ
まで[illegible]16進[illegible]だった

のです！

このように、メモリ上の１つの記憶場所には、００Ｈ～ＦＦＨまでの数値を記憶することができるのです。

[ビットとバイト]

「どうして、16進数なんて
わかりにくい[illegible]を使う必要がある[illegible]だ？」

ごもっともです。

皆さん、ご存知のようにコンピュータは０と１の電気信号しか理解できません。

ですから、コンピュータが直接理解できるのは

1011 0010 1000 1111

といったような

２進数

しか理解できないのです。

この２進数の１桁のことを

と呼びます。

ですから、４ビットで表わすことのできる最大の値は

１１１１

となるわけです。

また、このビットが8個集まったものを

1バイト（BYTE）

と呼びます。

1ビット			
8ビット	1バイト		
16ビット	2バイト	1ワード	
32ビット	4バイト	2ワード	1ダブル・ワード

さて、なぜ16進数を使うのかというと

2進数から16進数に変換しやすい

からなのです。

2進数の

0000　～　1111

を、16進数に変換すると

0H～FH

というように、2進数の4桁がちょうど16進数の1桁に変換できるわけです。

ところが、10進数に変換すると

0～15

というように、ハンパな値になってしまうのです。

2進数	16進数	10進数
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7

2進数	16進数	10進数
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

マシン語では、この

ビット

バイト

というのは、非常に重要な、データを扱う場合の単位ですから覚えておいてください。

3・3　レジスタとは

マシン語には

[illegible]

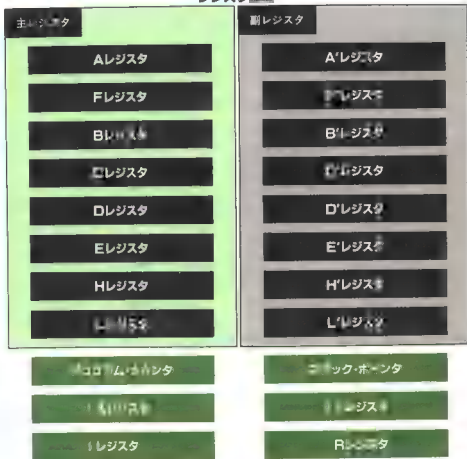
というものがあります。

レジスタは、命令を実行したり、いろいろな演算を行う時にデータや数値を一時記憶しておく場所のことです。

要するに

[illegible]

なのです。

あくまで――ようなもの――で、同じものというわけではありません。

レジスタと変数の異なる点は、

●レジスタの数は決まっており
変数のように増やすことはできない

●レジスタの扱える範囲は
1バイト（00H〜FFH）までである

●レジスタによって
それぞれ役割が違う

といったところです。

これらをキチンと理解していれば

レジスタ　変数

と考えても、ほとんど不都合な点はありません。

例えば、マシン語の

```
LD      A,0ECH
```

という命令は、BASICの

```
A = &HEC
```

といったカンジです。

【主レジスタと副レジスタ】

前ページの図を見ると

主レジスタ
副レジスタ

というのがありますね。

主、副といっても、機能が副レジスタよりも主レジスタの方が優れているわけではないのです。

ただ、我々が

同時に使えるのは、主レジスタのみ

なので、そう呼ばれるのです。

そして

EXX

などのエクスチェンジ命令を使って、これらの主レジスタと副レジスタの内容を交換することができるのです。

マア実際に使うのは、ほとんど主レジスタのみですから

副レジスタというものがある

と頭の片スミにでも入れておくだけで結構です。

【アキュームレータ】

一般に、Aレジスタのことを

アキュームレータ（Accumulator）

と呼び、他のレジスタと区別しています。

このように、Aレジスタを――アキュームレータ――と呼び他のレジスタと区別するのを、

演算の大部分がアキュームレータで行われる

からなのです。

【汎用レジスタ】

Bレジスタ・Cレジスタ
Dレジスタ・Eレジスタ
Hレジスタ・Lレジスタ

のことを

汎用レジスタ

と呼びます。

8ビットCPUであるμPD780C－1（Z80）は

1つのレジスタで は
1バイト（8ビット）しか扱えない

ように設計されています。

1バイトで表わすことのできる数は

00H～FFH（0～255）

しかないため、1バイトしか扱えないのでは非常に不便です。

ましてアドレスは、

F300H　EA58H　1B7EH

というように、2バイトで表わされているのですから、1つのレジスタでアドレスを指定することは不可能です。そこでレジスタは、

Bレジスタ	Cレジスタ
Dレジスタ	Eレジスタ
Hレジスタ	Lレジスタ

というように対（つい）にして

16ビット・レジスタ

として扱うこともできるようになっています。

3・4　フラグとは何か

先ほどの図のアキュームレータ（Aレジスタ）の下に

Fレジスタ

というのがありましたね？

このFレジスタは，他のレジスタとは、かなり違った役割をしています。

他のレジスタは、レジスタ自身の1バイトで1つの役割を持っていました。

ところが、このFレジスタは1バイトではなく、ビット単位で役割を持っているのです。

Fレジスタをビット単位（2進数）で表わすと、

S → 符号フラグ
Z → ゼロ・フラグ
H → ハーフキャリー・フラグ
P／V→ パリティ／オーバーフロー・フラグ
CY → キャリー・フラグ

となります。

これらのビットは

演算の結果によって

1になったり、0になったりする

わけです。

これを

フラグ

と呼びます。

このフラグは、実はマシン語で

条件判[illegible]

を行う時に使う、ひじょーうに重要なものなのです。

それは、もう少し後でくわーしく説明いたしますから、ここでは

どんな時にフラグが変化するか

という事を中心に説明することにしましょう。

【キャリー・フラグ（CYフラグ）】

キャリー・フラグは

演算の結果、桁上がりがあると　1
そうでなければ　0

または

減算の結果、桁借りがあると　1
そうでなければ　0

となります。

●桁上がり

CY=1

```
     1 0 0 0 0 0 1 1
+)   1 0 0 1 1 0 0 0
--------------------
[1]  0 0 0 1 1 0 1 1
```

CY=0

```
     0 1 0 0 0 0 1 1
+)   1 0 0 1 1 0 0 0
--------------------
[0]  1 0 0 1 1 0 1 1
```

●桁借り

CY=1

```
[1]  0 0 0 1 0 1 1 0
-)   0 0 0 1 0 1 1 1
--------------------
     1 1 1 1 1 1 1 1
```

CY=0

```
     1 0 0 1 0 1 1 0
-)   0 0 0 1 0 1 1 1
--------------------
     0 1 1 1 1 1 1 1
```

ですから、

演算の結果がＦＦＨ以上になった場合
減算で、引く方が引かれる方より大きかった場合

に

ＣＹ＝１

になるわけです。

【ゼロ・フラグ（Zフラグ）】

ゼロ・フラグは

[illegible]の結果が０になったら　１
そうでなければ　　　　　　　　０

となります。

演算の結果が０にな[illegible]

というと、減算で

１０Ｈ－１０Ｈ＝０

となるのを思いだしますが、加算でも

ＦＦＨ＋０１Ｈ＝０

となるのを、忘れないようにしてください。

●減算

●加算

3・5 LD命令

ここまで読んでこられた皆さんは

マシン語の基礎知識

については、ほぼ理解されたことと思います。

さて今度は

マシン語の命令

です。

とはいっても、全ての命令を説明するわけにはいきません。

ですから

必要な命令

だけ説明することにいたしましょう。

実際、マシン語には、あまり必要でない命令が多いのです。

【LD命令】

ＬＤ命令は

> **レジスタやメモリとの間で**
> **数値の受け渡しをする命令**

です。

例えば

```
LD        A,0ECH    Aレジスタに ECH を代入
```

とすると

アキュームレータにＥＣＨが代入される

わけです。

このＬＤ命令には、おもに

●レジスタの内容を任意のレジスタへ代入

→ LD A, B

●レジスタに任意の数値を代入

→ LD A, 0ECH

→ LD HL, 0ABCDH

●任意のメモリ内容をレジスタに代入

→ LD A, (HL)

→ LD A, (0F302H)

→ LD HL, (0F300H

Aレジスタの内容を任意のメモリに代入

→ LD (HL), A

→ LD (0EA58H), A

→ LD (0F300H), DE

というものがあります。

ＬＤ命令は、マシン語の命令の中で

最も期本的な命令

ですから、よく理解しておいてください。

【またしても実験!】

さて、せっかく皆さんは

マシン語の命令を１つ覚えた

のですから、このＬＤ命令を使ってカンタンなマシン語のプログラムを作ってみることにしましょう。

```
アドレス マシン語          ORG   0B900H          B900Hから作る
                   ;
B900  3AC8F3              LD    A,(0F3C8H)      AにF3C8Hの内容を代入
B903  32CAF3              LD    (0F3CAH),A      F3CAHにAを代入
B906  3E00                LD    A,0             Aレジスタに0を代入
B908  32C8F3              LD    (0F3C8H),A      F3C8Hに0を代入
                   ;
B90B  3E04                LD    A,4
B90D  D331                OUT   (31H),A         モニタへ戻る
B90F  CD4760              CALL  6047H
```

ORG 0D000H

JP 5C66H

という。まだ知らない命令がありますが、これは気にしないで結構です。

さーてさて、このプログラムが何をやっているかわかりますか?

「ウーム、Ｆ３Ｃ８Ｈ番地のメモリの内容を

Ｆ３ＣＡＨ番地のメモリに移して、それから

Ｆ３Ｃ８Ｈ番地のメモリに０を代入しているんだな」

さすが。その通りです!

確かにその通りなのですが、実は

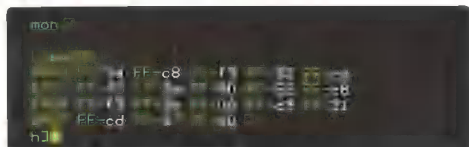
重大な秘密

がこのプログラムには隠されているのです。

その重大な秘密を知るには、

実際にプログラムを動かしてみるしかない

のです。

　まず

というふうにプログラムをキーインします。

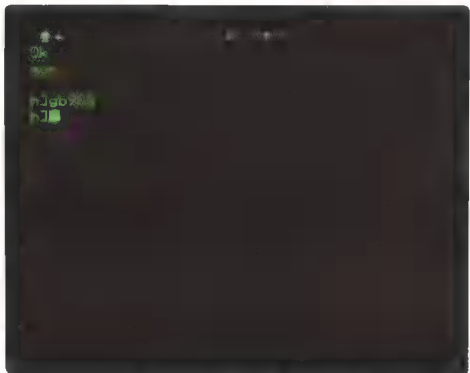
　次に、

　　　CTLR + B

でBASICへ戻って、

```
WIDTH 40 :CONSOLE 0,25,0,0
LOCATE 0,0 :PRINT "♠"
```

とキーインして画面設定を行ってから、プログラムを走らせます。

　すると、世にも恐ろしいことが起きたのです！　ジャジャーン!!

いやあ、ちょっとおおげさだったようですね。

「おい、ちっとも変っとらんではないか／」

よーく見てください。

どうです？

スペードが右に動いた

のがわかりますね。

コレが、あの

世にも恐ろしいこと

だったんですね、ハア。

3·6　演算命令

さて，前章の説明で

ＬＤ（ロード）命令

が

レジスタやメモリとの間で
数値の受け渡しをする命令

だということが皆さんにわかっていただけたと思います。

しかし、

数値の受け渡し

だけでは、

加算
減算
乗算
除算

などの演算を行うことはできませんね。

そこで、今回は

マシン語の演算命令

について説明することにいたしましょう。

【INC(インクリメント)命令、DEC(ディクリメント)命令】

前にも説明しましたように、

マシン語の命令は非常に細かい

のです。

その代表的な細かい命令が、この

ＩＮＣ(インクリメント)命令、ＤＥＣ(ディクリメント)命令

です。

ＩＮＣ命令は

レジスタの値に１を加える

INC命令

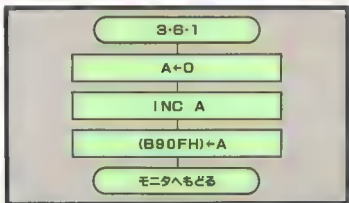
という命令です。

また、ＤＥＣ命令は

レジスタの値から１を引く

DEC命令

という命令です。

例えば

【ADD命令・SUB命令】

レジスタの内容に
1を加えたり、引いたりする

命令が

INC命令、DEC命令

でしたね。
ですから、この命令を使えば

マシン語で加算、減算ができる

わけです。
マア、4を加える程度でしたら

```
INC     A
INC     A
INC     A
INC     A
```

とすれば良いのですが、コレが

50を加える

なんてことにでもなったら

```
INC     A ┐
INC     A │
INC     A │
          ├─50コ
INC     A │
INC     A │
INC     A ┘
```

タイヘンですね。
そこで、マシン語には

ADD命令、SUB命令

という命令が用意されているわけです。なるほど、なるほど。
ADD命令は

レジスタの値に任意の数値を加える

ADD命令

という命令で、おもに

■**レジスタの内容に他のレジスタの値を加える**

→ ADD A, [illegible]
→ ADD HL, DE

■**レジスタの内容に任意の数値を加える**

→ ADD A, 1

→ ADD HL, 20H

■レジスタの内容に任意のメモリの内容を加える

→ ADD A, (HL)

という命令があります。

ＳＵＢ命令は

SUB命令

レジスタの値から任意の数値を引く

という命令で、おもに

●アキュームレータの内容から他のレジスタの値を引く

→ SUB B

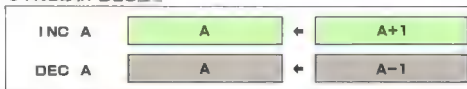
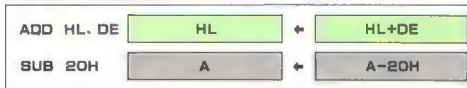
●アキュームレータの内容から任意の数値を引く

→ SUB 20H

■アキュームレータの値から任意のアドレスの内容を引く

→ SUB (HL)

があります。

●INC命令、DEC命令

●ADD命令、SUB命令

これで

ＩＮＣ命令、ＤＥＣ命令

ＡＤＤ命令、ＳＵＢ命令

についてわかってもらえたと思いますが、どうでしょうか？

ここで、これらの命令を使ったカンタンなプログラムをお見せいたしましょう。

どんなプログラムなのか？

は、皆さん方でお考えなってみるのも良いのではないでしょうか？

それが、マシン語マスターへの一番の近道なのですから。

```
                 ; ****************************
                 ; *    DEMO ﾌﾟﾛｸﾞﾗﾑ NO.0006    *
                 ; *        for PC-8801        *
                 ; *             by K.ﾂｶﾓﾄ     *
                 ; ****************************
                 ;
                       ORG   0B900H
                 ;
B900 21C8F3            LD    HL,0F3C8H
                 ;
B903 46                LD    B,(HL)
                 ;
B904 23                INC   HL
B905 23                INC   HL
                 ;
B906 7E                LD    A,(HL)
                 ;
B907 80                ADD   A,B
                 ;
B908 23                INC   HL
B909 23                INC   HL
                 ;
B90A 77                LD    (HL),A
                 ;
B90B 3E04              LD    A,4
B90D D331              OUT   (31H),A
B90F CD4760            CALL  6047H
```

3・7　JP(ジャンプ)命令とCP(コンペア)命令

マシン語の

重要な命令

の中に、

JP（ジャンプ）命令

があります。

この命令は、その名の通り

JP命令

指定したアドレスへジャンプする

という

BASICのGOTO命令に似た命令

です。

このJP（ジャンプ）命令には

●無条件JP命令

●条件JP命令

の2つがあります。

【無条件JP命令】

この無条件JP命令は

無条件JP命令

無条件に
指定したアドレスへジャンプする命令

です。

ですから

JP　　　5C66H

とすれば、もう、ただひたすら5C66H番地へジャンプするわけです。

【条件JP命令】

無条件JP命令が

無条件に
指定したアドレスへジャンプする

わけですから、条件JP命令は

条件によって
指定したアドレスへジャンプする

のではないでしょうか？

そう、その通りです。
条件ＪＰ命令は

> **決められた条件によって**
> **指定したアドレスへジャンプする命令**

条件ＪＰ命令

なのです。
しかし、その

条件

とはどのようなものなのでしょうか？

この条件ＪＰ命令に力を発揮するのが、前に説明した

フラグ

なのです。
フラグは、

演算の結果によって
０になったり
１になったりする

ものでしたね。
ですから、

フラグが１か０かを見れば
演算の結果がわかる

わけです。
フラグには

7	6	5	4	3	2	1	0
S	Z	＼	H	＼	P／V	N	CY

Ｓ　➜　符号フラグ
Ｚ　➜　ゼロ・フラグ
Ｈ　➜　ハーフキャリー・フラグ
Ｐ／Ｖ➜　パリティ／オーバーフロー・フラグ
Ｎ　➜　加／減算フラグ
ＣＹ　➜　キャリー・フラグ

という６種類がありますが

実際に使われるのは
ゼロ・フラグがほとんど

ですから、今回は

ゼロ・フラグを使った条件ＪＰ命令

について説明することにいたしましょう。

〔ゼロ・フラグを使った条件JP命令〕

ゼロ・フラグは

　　演算の[illegible]が0ならば　　1
　　そうでなければ　　0

というふうになりましたね。

ですから、このゼロ・フラグの値によって

　　●ゼロ・フラグが1であればジャンプ
　　　→ JP　Z，アドレス
　　●ゼロ・フラグが0であればジャンプ
　　　→ JP　NZ，アドレス

という2種類の条件JP命令が用意されています。

Z＝1の場合	JP　　Z，アドレス
Z＝0の場合	JP　　NZ，アドレス

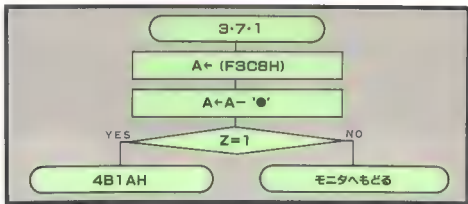
さて、この条件JP命令を使って

　　ちょっとしたプログラム

を作ってみることにいたしましょう。

条件JP命令を使うと面白いプログラムを作ることができるのです。

サテ、面白いプログラムのはじまり、はじまり。

```
                ; ******************************
                ; *  ﾁｮｯﾄ ｼﾀ DEMO ﾌﾟﾛｸﾞﾗﾑ      *
                ; *         for PC-8801        *
                ; *                by K.ﾂｶﾓﾄ *
                ; ******************************
                ;
                        ORG   0B900H            B900Hから作る
                ;
B900 3AC8F3             LD    A,(0F3C8H)
                ;
B903 D6EC               SUB   '●'
B905 CA1A4B             JP    Z,4B1AH
                ;
B908 3E04               LD    A,4
B90A D331               OUT   (31H),A
B90C CD4760             CALL  6047H
```

このプログラムは、

●[illegible]の左上に〈●〉があった場合

BASICに戻る

●画面の左上に〈●〉がなかった場合

モニタに戻る

という内容のプログラムです。

『本当にそうなるのか？』

とにかく実際に動かしてみましょう。

まず

```
WIDTH 40,25
CONSOLE 0,25,0,0
```

とキーインして画面設定を行います。

次に

とキーインして、マシン語プログラムを入力します。

サテ、まずは

画面の左上に〈●〉がある場合

の方から試してみることにいたしましょう。

ヤヤッ！　ちゃんとBASICに戻ったようですね。

今度は

画面の左上に〈●〉がない場合

を試してみることにいたしましょう。

どうです？　うまく動きましたね。

さてさて，どうしてこのような動作をするのでしょうか？

まず

[illegible]

とすると，

アキュームレータに画面の
左上のキャラクタの値が代入

されます。

ですから、画面の左上に〈●〉が表示されている場合には、アキュームレータにＥＣＨが代入されるわけです。

次に

[illegible]

として

[illegible]

ます。

ですから、アキュームレータの値がＥＣＨであった場合、結果は０となりゼロ・フラグが１となるわけです。

このフラグを見て

[illegible]

で、条件ＪＰ（ジャンプ）しているわけです。

【CP命令】

先ほどの説明で

```
SUB     'ⓔ'
JP      Z,4B1AH
```

とすると、

■アキュームレータがＥＣＨの場合００８１Ｈ番地へ
■アキュームレータがＥＣＨでない場合モニタへ戻る

というように

条件ジャンプ

できることをわかっていただけたと思います。

ところが、この方法だと実際にアキュームレータから引くのですから、アキュームレータの値が変わってしまい不便です。

条件ジャンプに必要なのは、

引いた時のフラグの変化

で、アキュームレータから実際に引く必要はないのです。

いや、実際に引いてもらっては困るのです。

どこかに

フラグだけ変化する減算

がないでしょうか？

実は、あるのです。

フラグだけ変化する減算　　**CP命令**

それが

ＣＰ（コンペア）命令

なのです。

ですから、先ほどのプログラムは

```
                      ORG   0B900H
                ;
B900 3AC8F3           LD    A,(0F3C8H)
                ;
B903 FEEC             CP    'ⓔ'          ← Aレジスタの内容とECHを比較する
B905 CA1A4B           JP    Z,4B1AH      ← Z=1ならBASIC
                ;
B908 3E04             LD    A,4
B90A D331             OUT   (31H),A         モニタ
B90C CD4760           CALL  6047H
```

とすれば良いわけです。

3・8 CALL命令とRET命令

先ほどのプログラム

```
                    ; ******************************
                    ; * ﾁｮｯﾄ ｼﾀ DEMO ﾌﾟﾛｸﾞﾗﾑ      *
                    ; *         for PC-8801        *
                    ; *              by K.ﾂｶﾓﾄ *
                    ; ******************************
                    ;
                            ORG   0B900H
                    ;
B900  3AC8F3                LD    A,(0F3C8H)
                    ;
B903  D6EC                  SUB   '●'
B905  CA1A4B                JP    Z,4B1AH
                    ;
B908  3E04                  LD    A,4
B90A  D331                  OUT   (31H),A
B90C  CD4760                CALL  6047H
```

では

画面の左上しか調べなかった

ですね。

そこで今回は

画面の左上と右上を調べる

プログラムを作って見ることにいたしましょう。

```
                   ;  ■*****************************
                   ;  ■  オモシローイ  DEMO プログラム      ■
                   ;  ■         for PC-8801          ■
                   ;  *                  by K.ツカモト ■
                   ;  ■*****************************
                   ;
                          ORG   0B900H
                   ;
B900 21C8F3               LD    HL,0F3C8H
                   ;
B903 7E                   LD    A,(HL)
B904 FEEC                 CP    '●'
B906 CA1A4B               JP    Z,4B1AH
                   ;
B909 2116F4               LD    HL,0F416H
                   ;
B90C 7E                   LD    A,(HL)
B90D FEEC                 CP    '●'
B90F CA1A4B               JP    Z,4B1AH
                   ;
B912 3E04                 LD    A,4
B914 D331                 OUT   (31H),A
B916 CD4760               CALL  6047H
```

まずは

とキーインしてプログラムを入力してください。

間違いはありませんか？

確認してください。

もし間違いがありましたら，もう一度キーインしなおしてください。

3·8·1
LD HL, 0F3C8H
LD A, (HL)
CP '●'
Z=1
YES
4B1AH
NO
LD HL, 0F416H
LD A, (HL)
CP '●'
Z=1
YES
4B1AH
NO
モニタへもどる

さてさて、

実際にプログラムを動かしてみましょう。

まずは

右上に〈●〉を表示しておいて試す

ことにいたしましょう。

ムムムム、ちゃんとBASICに戻りましたね。

次は、

左上に〈●〉を表示しておいて試してみる

ことにいたしましょうか。

うーむやはりBASICに戻った

ウーム、やはり

BASICに戻った

ようです。

コレで

プログラムが正常に作動している

ことは

確認できた

わけです。

さて、ここで先ほどのプログラムをもう1度見てみることにいたしましょう。

```
                ; *■■■*****■*****■■■*************
                ; ■  オモシローイ  DEMO プログラム      *
                ; *           for PC-8801          *
                ; *                  by K.ツカモト ■
                ; ■■■***■■*■■*■■■*****************
                ;
                        ORG   0B900H
                ;
B900 21C8F3             LD    HL,0F3C8H
                ;
B903 7E                 LD    A,(HL)
B904 FEEC               CP    '●'
B906 CA1A4B             JP    Z,4B1AH
                ;
B909 2116F4             LD    HL,0F416H
                ;
B90C 7E                 LD    A,(HL)
B90D FEEC               CP    '●'
B90F CA1A4B             JP    Z,4B1AH
                ;
B912 3E04               LD    A,4
B914 D331               OUT   (31H),A
B916 CD4760             CALL  6047H
```

「お！　プログラムの中に

同じ所が２か所ある！。」

さすが。注意深い皆さんは、お気付きになられたようですね。

そうなのです。

このプログラムには、

同じ所が２か所ある

のです。

同じ所が２か所あるとは

もったいない

ですね。どうにかならないものでしょうか？

実は、どうにかなるのです。

そのために用意されている命令が

ＣＡＬＬ（コール）命令

ＲＥＴ（リターン）命令

なのです。

ＣＡＬＬ命令は

CALL命令

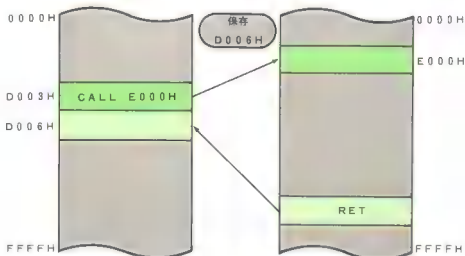
次の命令のアドレスを保存して 指定されたアドレスへジャンプする命令

です。

また、ＲＥＴ命令は

RET命令

保存してあるアドレスへ戻る命令

です。

```
                ; ***********■*****************
                ; *  オモシローイ  DEMO プログラム      ■
                ; *           for PC-8801             *
                ; *                       by K.ツカモト *
                ; ****************************■**
                ;
                        ORG   0B900H       B900 から作る
アドレス マシン   ;
B900 21C8F3             LD    HL,0F3C8H    HLにF3C8Hを代入
B903 CD13B9             CALL  SUB          SUBサブルーチンをコー■
                ;
B906 2116F4             LD    HL,0F416H    HLにF416Hを代入
B909 CD13B9             CALL  SUB          S ∃B サブルーチンをコール
                ;
B90C 3E04               LD    A,4
B90E D331               OUT   (31H),A      モニタに戻る
B910 CD4760             CALL  6047H
                ;
                ;
                ; *****  サブルーチン  *****
                ;
B913 7E         SUB:    LD    A,(HL)
B914 FEEC               CP    '●'
B916 CA1A4B             JP    Z,4B1AH      サブルーチ
                ;
B919 C9                 RET
```

このように

CALL命令

RET命令

を使うと

●プログラムが短かくなる

●プログラムが見やすくわかりやすくなる

という効果が表われてきます。

また、このように

CALL命令

RET命令

を使うことを

サブルーチン化

といいます。

【この章のおわりに】

皆さん、この

[illegible]

どうでしたでしょうか？

たったの40ページ

で、マシン語の用語・命令を説明するのですから、かなりムリがありますが、

重要な命令

についてはわかっていただけたのではないでしょうか？

さて、次の章は、おまちかね

[illegible]

です。

第4章

内部サブルーチンの活用

4·2 アセンブラのための命令

これから

内部ルーチンを活用

していくわけですから

数々のマシン語プログラム

を

アセンブラ

で作らなければなりません。

アセンブラというものは

人間がわかりやすいニーモニック
で組んだプログラムを
マシン語に訳すプログラム

のことでしたね。

実は、命令の中に

アセンブラを扱うための命令

があるのです、ハイ。

「ど、どれなんだ。その命令は」

マアマア、あせらないでください。

内部ルーチンを説明する前に、まず、この

アセンブラを扱うための命令

について説明することにいたしましょう。

【ORG命令】

ニーモニックで組んだプログラムをアセンブラがマシン語に訳してメモリ上に移す時

メモリのどこからにするか

を指定しなければなりませんね。

この

メモリのどこからにするかを指定する命令

ORG命令

が

なのです。

ここで前に作ったマシン語プログラムを見て、ORG命令の使い方を確認してみましょう。

```
                    ; *****************************
                    ; *  オモシローイ  DEMO プログラム      *
                    ; *          for PC-8801          *
                    ; *                by K.ツカモト *
                    ; *****************************
                    ;
                         ORG    0B900H      ← B900Hから作る
アドレス マシン語    ;
B900  21C8F3             LD     HL,0F3C8H   ← HLにF3C8Hを代入
                    ;
B903  7E                 LD     A,(HL)      ← AにHLの[illegible]の値を代入
B904  FEEC               CP     '●'           AとECHを比較
B906  CA1A4B             JP     Z,4B1AH     ← Z=1ならBASIC[illegible]
                    ;
B909  2116F4             LD     HL,0F416H   ← HLにF416Hを代入
                    ;
B90C  7E                 LD     A,(HL)
B90D  FEEC               CP     '●'
B90F  CA1A4B             JP     Z,4B1AH
                    ;
B912  3E04               LD     A,4
B914  D331               OUT    (31H),A       モニタに[illegible]
B916  CD4760             CALL   6047H
```

どうです？

ORG　　　0B900H

と指定してますから、ちゃんと

B90[illegible]からマシン語が入っている

ことがわかりますね。

【"0"は[illegible]字のマーク】

サテ、上のプログラムを見ると

0B900H ←	B900H
0F3C8H ←	F3C8H
0F416H ←	F416H

というふうに

なぜか[illegible]字の前に〈0〉がついている

のです。

ナゼ、数字の前に〈0〉がついているのでしょうか？

16進数は

10以上の数にA～Fのアルファベット
を使っている

わけでしたね。

ですから、極端な話、

FACEH　（face【名】顔）
ADAMH　（Adam【名】アダム）
BEACH　（beach【名】浜）

というような16進数も考えられるわけです。

これではまぎらわしくてしかたありません。

そこで

> **16[illegible]数の最初がアルファベット[illegible]場合**
> **〈0(ゼロ)〉をつける**

わけです。

【ラベルの[illegible]】

アセンブラには

と呼ばれる、ひじょ――に便利な命令があります。

『ラベルって何なんだ？』

マア、とにかく、その

ラベル

というものを見てやろうじゃありませんか！

```
GVRAMO:  LD          A,(HL)
```

コレが、その

ラベル

というものです。

ラベルは、その名の通り、名札のようなもので、

のです。

アセンブラには

行番号がない

のですから、ジャンプ先やコール先を指定する時困るわけです。

そこで！　ラベルが活躍するのです。

ラベルは、普通

[illegible]　（コロン）を伴って

6文字以下の英数字

で表わされます。

このラベルを命令の前に書くと、ラベルはその命令の先頭アドレスの値になるのです。

```
アドレス マシン語
                 ; ******************************
                 ; *  オモシローイ  DEMO プログラム     *
                 ; *           for PC-8801       *
                 ; *                 by K.ツカモト *
                 ; ******************************
                 ;
                          ORG   0B900H
                 ;
B900 21C8F3               LD    HL,0F3C8H
B903 CD13B9               CALL  SUB
                 ;
B906 2116F4               LD    HL,0F416H
B909 CD13B9               CALL  SUB
                 ;
B90C 3E04                 LD    A,4
B90E D331                 OUT   (31H),A
B910 CD4760               CALL  6047H
                 ;
                 ;
                 ; *****   サブルーチン   *****
                 ;                               SUBはB913Hの値になる
B913 7E          SUB:     LD    A,(HL)
B914 FEEC                 CP    '●'
B916 CA1A4B               JP    Z,4B1AH
                 ;
B919 C9                   RET
```

【EQU命令】

この非常に有効な

ラベルにユーザーが任意の値を設定する命令

が、

EQU命令

です。

例えば、

VRAMAD: EQU 0F300H

というように、EQU命令を使うと

VRAMAD

というラベルは、

0F300H

の値になるわけです。

サテここで、この

ラベル

を使ったマシン語プログラムを紹介いたしましょう。

「どんなプログラムなのか？」

それは、皆さん、自分自身で調べてみてください。

```
                    ; **********************************
                    ; *   CRT ラ ´●´ デ" ウメル プ°ログ"ラム   *
                    ; *          for PC-8801              *
                    ; *                   by K.ツカモト    *
                    ; **********************************
                    ;
F3C8                VRAMAD:EQU   0F3C8H
0028                ADD40: EQU   40
                    ;
                           ORG   0B900H
アドレス マ
                    ;
B900 21C8F3                LD    HL,VRAMAD     HLにVRAMADの値を代入
B903 112800                LD    DE,ADD40      DEにADD40の値を代入
                    ;
B906 0E19                  LD    C,25
B908 0650           LOOP1: LD    B,80
B90A 36EC           LOOP2: LD    (HL),´●´
B90C 23                    INC   HL
B90D 10FB                  DJNZ  LOOP2
B90F 19                    ADD   HL,DE         HL=HL+DE
B910 0D                    DEC   C
B911 20F5                  JR    NZ,LOOP1
                    ;
B913 3E04                  LD    A,4
B915 D331                  OUT   (31H),A
B917 CD4760                CALL  6047H
```

[データ[illegible]命令]

アセンブラを扱う命令の中に

BASICのＤＡＴＡ文に似た命令

まずは、アセンブラを使ってプログラムを

というふうにキー・インします。

おっと！　アセンブラによっては

プログラムの打ち込みが違う

ので注意してください。

キー・インし終りましたら、お持ちのアセンブラのマニュアルを見て

してください。

アセンブルすると

となります。

アセンブラのマニュアルを見て、できたマシン語を

メモリに移してください。

「おーい、■だアセンブラ持ってないぞ」

ご愁傷さまデス。まだアセンブラを購入されていない方は

とキー・インしてください。

このように、マシン語プログラムをメモリ上に作りましたら、今度は

実行

することにいたしましょう。

```
mon

hJgb900
```

とすると

```
How many files(0-15)? 0
NEC N-88 BASIC Version 1.0
Copyright (C) 1981
Ok
```

というように

　　　　リセット

されました。

　さて、先ほどの内部ルーチンの概要には

説　　　明	N－BASICのイニシャライズを行い、ストップ・キーが押されていればホット・スタートへ。押されていなければコールド・スタートへジャンプします。

と書かれてありましたね。
　皆さん、よーく見てください。

　　　　[illegible]
　　　　[illegible]

という、今まで聞いたことのない単語がでてきましたね。
　これらの単語は、いったいどのような意味なのでしょうか？
　実は、コールド・スタートというのは

> **電源をONにした状態のように**
> **全ての初期設定を行うこと**

なのです。
　ですから、先ほど皆さんに

　　　　[illegible]

してもらったのは

　　　　コールド・スタートを行った

ことになるのです。
　それでは、ホット・スタートというのはどのような意味なのでしょうか？
　ホット・スタートはウォーム・スタートとも呼ばれ

> **何らかの原因によりマイコンが暴走した場合に**
> **必要な部分のみを初期設定し**
> **正常な状態に戻すこと**

を意味するのです。

　さてここでもう１度、先ほどの内部ルーチンの概要を見てみることにいたしましょう。

内部ルーチンの概要には

……ストップ・キーが押されていれば

ホット・スタートへ……

というように書いてありますね。

ですから、

わけです。

【STOP押してホット・スタート】

ここまでわかれば、しめたものです、ハイ。

さっそく

実験！

してみることにいたしましょう。

皆さんのパソコンには、

先ほどキー・インしたマシン語プログラム

が入っていますね。

『あんなのもう消しちゃった！』

このような方は、ご面倒ですがもう１度キー・インしてください。

これで、皆さんのパソコンには先ほどのマシン語プログラムが入ったわけです。

それではまず、

```
mon
*J9B900
```

とキー・インします。

おっと！　まだ(RETURN)キーは押さないでくださいね。

このようにキー・インしましたら

にしてください。

「右手の方がいいですか、それとも左手ですか？」

どっちでもいいですよ。たいして変りませんから――。

そうしましたら、おもむろに

(RETURN)キーを押したか押さないかわからないくらいのスピ

ードで

するど、画面は

というようになります。

すなわち

わけです。

(RETURN)キーを押した瞬間(STOP)キーを押さなければならないので、うまくホット・スタートされない方は、もう少し速く(STOP)キーを押すように心掛け、もう1度チャレンジしてみてください。

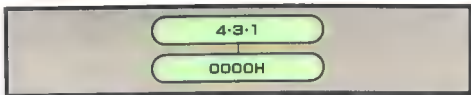
『おーい何回トライしてもダメだぞー!』

このような方は、少々運動神経がにぶいようですので、マァ、あきらめていただくしかないようです。

そうそう

先ほどのプログラムのフローチャート

を忘れておりました。

これが先ほどのプログラムのフローチャートです。

この

PC－8801のリセットを行う内部ルーチン

は

一見たいしたことのない

ように思ってしまうかもしれません。

が、しかし、この内部ルーチンこそ

内部ルーチンの基本

なのです。

考えてみてください。

JP　　　0000H

この、たったひとつの命令で

- **●ストップ・キーを押しているか**
- **●画面の初期化**
- **●テキスト・変数の初期化**
- **●ワーク・エリアの初期化**

など、多くのことを行っているのです。

何はともあれ、皆さんは

内部ルーチン活用の第1歩

を踏みだしたのです！

4・4　マシン語からBASICへ

前回の章で

ＰＣ－８８０１をリセットする内部ルーチン

を活用しましたね。

皆さんは

初めて内部ルーチンを活用した／

と思ったのではないでしょうか？

ところが！　ところがなのデス。

実は、皆さんは、この――リセットする内部ルーチン――の他に

すでに『[illegible]用していた』

のです。

『そんな覚えはないゾ』

ごもっともです。皆さんは

気づかないうちに活用していた

のですから――。

前に

```
                        ORG   0B900H
                ;
B900  3AC8F3            LD    A,(0F3C8H)
                ;
B903  FEEC              CP    '●'
B905  CA1A4B            JP    Z,481AH
                ;
B908  3E04              LD    A,4
B90A  D331              OUT   (31H),A
B90C  CD4760            CALL  6047H
```

というマシン語プログラムを作ったのを覚えていらっしゃるでしょうか？

このプログラムは

> **画面の左上に〈●〉があったらＢＡＳＩＣへ**
> **なければモニタへ戻る**

という内容のプログラムでしたね。思い出してください。

実は、このプログラムですでに内部ルーチンを活用していたのです。

さて、このプログラムの中に

JP Z,4B1AH

という命令があります。

この命令だけの意味は

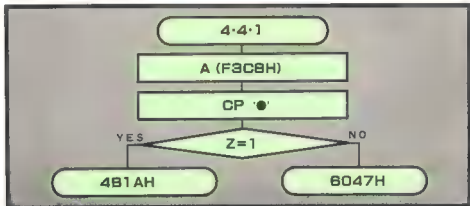
Z（ゼロ）フラグが１であれば ４Ｂ１ＡＨ番地へジャンプする

というものですね。

マシン語の場合

のですから、

プログラム全体の構造

を見ることにいたしましょう。

フローチャートを見ておわかりの通り、この

JP Z,4B1AH

をプログラム全体から見ると

画面の左上に〈●〉があった場合 ００８１Ｈ[illegible]へジャンプする

という意味になるのです。

【0081Hは殺しの番号】

しかし、００８１Ｈ番地へジャンプすると

なぜＢＡＳＩＣへ戻るのか？

それは

[illegible]

が、

BASICへ戻るための内部ルーチン

だからなのです！

4B1AH：N88－BASICのコマンド待ちを行う

アドレス	4B1AH
機能	N88－BASICのコマンド待ちを行う
レジスタ	―――
引数	なし
説明	イニシャライズをせず、[illegible]にプロンプトである「OK」を表示し、N88－BASICのスクリーン・エディト・モードに入る。 このアドレスにジャンプすることによりBASICに制御を移すことが可能です。

さて実際に

この内部ルーチンを活用

することにいたしましょう。

```
                ; ***************************
                ; * 4B1AH: N88-BASIC ヘ モト゛ル *
                ; *          for PC-8801     *
                ; *             by K.ツカモト *
                ; ***************************
                ;
                        ORG    0B900H
                ;
B900 C31A4B             JP     4B1AH
                ;
```

4·4·2 ― 4B1AH

このプログラムを

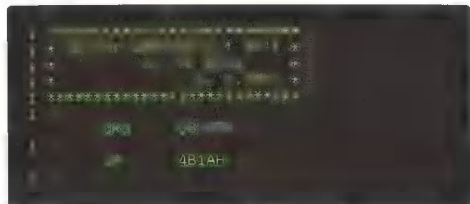
アセンブラをお持ちでない方

が入力する場合は次のようにキーインしてください。

```
h]gb900
8900 [illegible]
h]
```

また

アセンブラをお持ちの方

は、

というようにキー・インしてからアセンブルを行い、メモリにマシン語プログラムを移してください。

アセンブラでは、このような

ニーモニックで表されたプログラム

を

と呼び、また、

変換されて作られたマシン語プログラム

を

と呼びます。

おっと！　少し脇道にそれてしまいましたね。

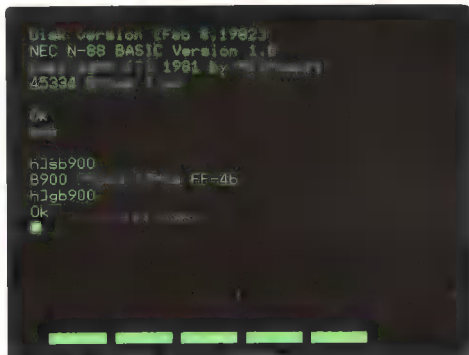
このように、メモリにマシン語プログラムを作りましたら

```
h]gb900
```

とキー・インして、

プログラムを実行

してください。

すると画面は、

というふうにな!!

ＢＡＳＩＣへ戻った／

わけです。

4・5 マシン語からモニタへ

前節の初めで、皆さんは

すでに2つ内部ルーチンを活用していた

と、ワタクシは説明いたしました。

```
                          ORG   0B900H          B900Hから作る
アトレス  マシン語    ;
B900  3AC8F3              LD    A,(0F3C8H) ←── AにF3C8Hの内容を移す
                      ;
B903  FEEC                CP    '●'             AとECHを比較
B905  CA1A4B              JP    Z,4B1AH         Z=1ならBASICへ
                      ;
B908  3E04                LD    A,4
B90A  D331                OUT   (31H),A
B90C  CD4760              CALL  6047H
```

1つは

4B1AH　BASICへ戻る

ということは、皆さんにわかってもらえたと思います。

しかし

もう1つは一体何なのでしょうか?

[6047Hでモニタへ]

注意深い方は、すでにお気づきになっているかもしれませんが、すでに活用した内部ルーチンとは

6047H

にある

モニタへ制御を移す内部ルーチン

のことなのです。

6047H：マシン語モニタに制御を移す

アドレス	6047H
機能	マシン語モニタに制御を移す
レジスタ	―――――
引数	なし
説明	スタック・ポインタを再定義し画面にモニタのプロンプトである〝h]〟を表示した後、マシン語モニタに制御を移す。

しかし、困ったことに

BASICへ戻る

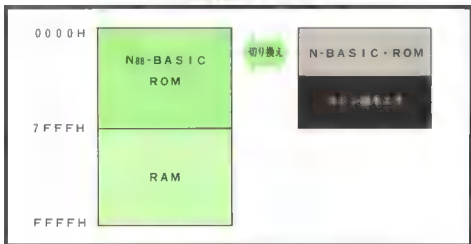
時のようにカンタンにはできないのです。

どうしてでしょうか？

それには、まず次の

PC-8801のメモリ・マップ

をみてください。

PC-8801のメモリ・マップ

さあ、わかっていただけたでしょうか？

そう、その通りです。

実は、PC-8801のN88-BASICとモニタは

別のROMに記憶されている

のです。

通常は

N88-BASICが用意されている

ので

モニタに戻る時は

モニタにあるROMをセレクトしなければならない

わけです。

このN88-BASICのみ記憶されているROMを

N88-BASIC·ROM

N-BASICとN88-BASICモードのモニタが記憶されている

ROMを

N-BASIC·ROM

と呼びます。

N-BASIC・ROMをセレクトするには

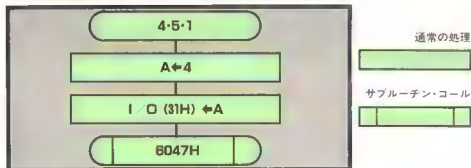
```
  LD        A,4
  OUT       (31H),A
```

とマシン語で実行します。

次に、先程の

モニタへ戻る内部ルーチンをコール

すればモニタに戻ることができるわけです。

```
                  ;  ****************************
                  ;  * 6047H:モニタ ヘ モドル プログラム *
                  ;  *         for PC-8801       *   ←注釈
                  ;  *              by K.ツカモト   *
                  ;  ****************************
                  ;
                        ORG   0B900H      ← B900Hから作る
                  ;
B900  3E04              LD    A,4
B902  D331              OUT   (31H),A     ← モニタへ戻る
B904  CD4760            CALL  6047H
```

このプログラムを

アセンブラをお持ちの方

が入力する場合は、次のようにキーインして

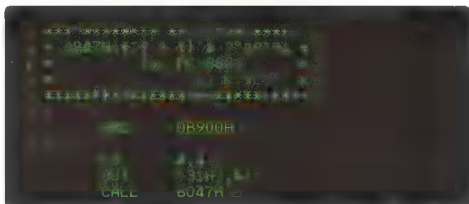
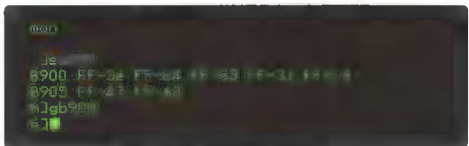
ソース・リスト（先ほど説明しましたね）

を作り、アセンブルを行い

オブジェクト（コレも先ほど説明しましたね）

に落とします。

いやあ、こういうふうに『ソース・リスト』とか『オブジェクト』という単語を使うと、いかにも

マシン■

4·6 WIDTHをマシン語で

マシン語でプログラムを組む時

　　　一番、手間がかかるもの

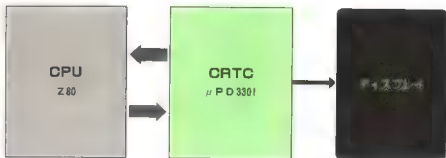
が

　　　画面の設定

といわれています。

なぜ、画面の設定が——一番手間がかかるのか——というと、画面の設定を行うにはマシン語の知識だけでなく

　　　ハードウェアの知識

もある程度必要だからなのです。

この図を見てわかるように

　　　実際にディスプレイ（テレビ）に
　　　信号を送っているのはＣＲＴＣ
　　　というマイクロ・プロセッサ

なのです。

ですから

　　　画面の設定を行うには
　　　ＣＲＴＣを扱わなければならない

わけです。

【画面設定も内部ルーチンを使えば】

このように、BASICの命令では

　　WIDTH 40,20

というふうに、またたく間にできる

　　　画面の文字数設定

も、いざ、マシン語で同じことを行おうと思うと

かなり難しい

わけです。

マシン語で組むと

画面設定は難しい――。

これは

すべて自分で組んだ場合

のこと。

自分で組めば難しい画面設定も

内部ルーチンを活用すれば

簡単にできる

のです。

【まずは80文字モードに】

画面の文字数を設定するには

６Ｆ６ＢＨ番地からの

画面モード設定内部ルーチン

を活用します。

この内部ルーチンを使えば

画面の文字数を設定できる

わけです。

それでは、その内部ルーチンの

概要

を見てみましょう。

6F6BH：画面のモード設定を行う

アドレス	６Ｆ６ＢＨ
機　能	画面のモード設定を行う
レジスタ	Ａ，Ｆ，Ｂ，Ｃ，Ｄ，Ｅ
引　数	Ｂ，Ｃ，その他ワークエリア
説　明	Ｂレジスタに[illegible]の桁数を入れ、Ｃレジスタに行数を入れて６Ｆ６ＢＨ番地を呼ぶことによって画面の文字数を設定する。 この時、同時にワークエリアによって他の画面モードも設定する。

この

６Ｆ６ＢＨ：画面のモード設定を行う

内部ルーチンを使って

画面を80桁、20行にしてやろう

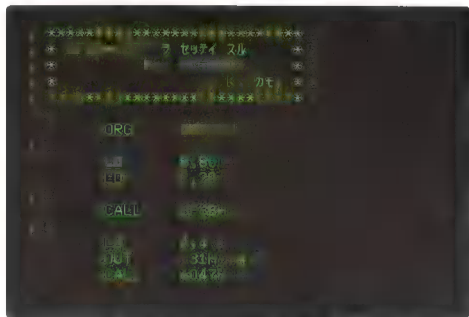
ではありませんか。

```
                        ; ****************************
                        ; * 6F6BH:CRT ヲ セﾂテイ スル     *
                        ; *          for PC-8801        *
                        ; *              by K.ツカモト   *
                        ; ****************************
                        ;
                                ORG   0B900H
                        ;
B900 0650                       LD    B,80
B902 0E14                       LD    C,20
                        ;
B904 CD6B6F                     CALL  6F6BH
                        ;
B907 3E04                       LD    A,4
B909 D331                       OUT   (31H),A
B90B CD4760                     CALL  6047H
```

このように

をおこない

Bレジスタに80を代入

Cレジスタに20を代入

```
How many files(0-15)? 0
NEC N-88 BASIC Version 1.1
Copyright (C) 1981 by Microsoft
45914 Bytes free

sb900
B900 FF-06 FF-50 FF-0e FF-19 FF-cd
B905 FF-6b FF-6f FF-3e FF-04 FF-d3
B90A FF-31 FF-cd FF-47 FF-60
B]
```

というようにキーインします。

これで、プログラムの入力は終りましたね。

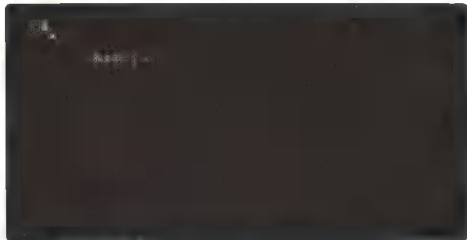
それでは、それでは

実行

してみることにいたしましょう。

さあ

gb900

とキーインしてください。

ややっ！ 画面が

80文字モード

になりましたね。

【今度は40桁に設定】

さて

80桁に設定

したのを

40桁に戻す

のを今度は実行してみることにいたしましょう。

とはいっても、プログラムの違いは

80桁	B900 0650	LD	B,80
40桁	B900 0628	LD	B,40

というように

たった1バイト

ですから、

メモリーにあるマシン語プログラムを

直接書き換える

ことにします。

80桁	B900 06 50 0E 19 CD 6B 6F 3E
40桁	B900 06 28 0E 19 CD 6B 6F 3E

上の図を見ておわかりの通り、書き換えるアドレスは

B901H

なわけですから

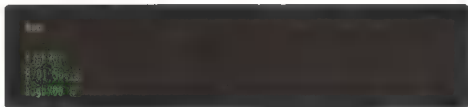
とキーインすれば

40桁に戻すプログラム

に変更することができるわけです。

これでプログラムが――40桁に戻すプログラム――になったのですから、このプログラムを実行して

画面を40桁にする

ことにいたしましょう。

すると、画面は

というように懐かしの40桁モードになりました。

さあ、これで皆さんは

[illegible]

をマスターしたのです。

【今[illegible]は行数の設定だ！】

皆さんは、すでに

画面の文字数を設定する内部ルーチン

をマスターしました。

しかし

画面のモード設定を行う内部ルーチン

を使っていながら、前回は

画面の桁数の設定

を行っただけでしたね。

そこで、今度は先程の内部ルーチンを使って

画面の行数の設定を行う

ことにいたしましょう。

とはいっても、前回使った内部ルーチンをおなじように活用するのですから難しく考えることはありません。

では、もう一度画面のモード設定を行うルーチンの概要を見てみましょう。

６Ｆ６Ｂ：画面のモード設定を行う

アドレス	６Ｆ６ＢＨ
機　能	画面のモード設定を行う
レジスタ	Ａ、Ｆ、Ｂ、Ｃ、Ｄ、Ｅ
引　数	Ｂ、Ｃ、その他ワークエリア

Ｂレジスタに画面の桁数を入れ、Ｃレジスタに行数を入れて６Ｆ６ＢＨ番地を呼ぶことによって画面の文字数の設定を行う。

この[illegible]、同時にワークエリアによって他の画面モードも設定する。

どうです？

どうやって、画面の行数を設定したらいいのか、皆さんもお気づきになられたと思います。

ですから

『これなら自分だけでも活用できそうだ！』

と思われた方は

今すぐ本書をとじて
自分で活用するプログラムを
作ってください。

『だいたいわかるのだが、少し不安だなァ』

このような方は、ひき続き本書を読んでワタクシとともに活用するプログラムを作っていってください。

あと一歩という所まで来ているのですから――。

【活用活用、また活用】

では、では

とやらを作んでみようではありませんか！

```
                 ; *************************
                 ; * 6F6BH:CRT ｦ ｾｯﾃｲ ｽﾙ     *
                 ; *          for PC-8801   *
                 ; *             by K.ﾂｶﾓﾄ *
                 ; *************************
                 ;
                          ORG   0B900H
                 ;
B900 0628                 LD    B,40
B902 0E19                 LD    C,25
                 ;
B904 CD6B6F               CALL  6F6BH
                 ;
B907 3E04                 LD    A,4
B909 D331                 OUT   (31H),A
B90B CD4760               CALL  6047H
```

実際にプログラムを見ると――画面の桁数を設定する内部ルーチンとほぼ同じということがよくわかりますね。

このように

のことを

といいます。

内部ルーチンの多くは、この

引数

を利用しているのです。

さて、前置きはこれくらいにして、実際に

活用

するといたしましょうか、ハァ。

このプログラムを

アセンブラをお持ちの方

が入力する場合は

とキーインします。

前にも説明しましたが――アセンブラにより入力方法が違うので、アセンブラのマニュアルをよく読んでからキーインしてください。

さて

アセンブラをお持ちでない方

が入力する場合は

```
B900, FF-0b FF-28 FF-0e FF-[illegible] FF-cd
B905, FF-6b FF-6f FF-3e FF-04 FF-d3
B90A, FF-31 FF-cd FF-47 FF-60
```

とキーインします。

これで

を入力し終えたわけです。

さァ、今度は、おまちかねプログラムを実行することにいたしましょう。

このプログラムは

B900H番地

から作られているのですから、このプログラムを実行するには

とキーインすればいいわけですね。

おっと！

アセンブラをお持ちの方

は、アセンブルを行い、入力したプログラムをオブジェクトに変換してからメモリに移してください。

この方法は、アセンブラによって異なりますから、マニュアルをよーくお読みになって行ってください。

一見― 難しそう――なこの作業も、慣れてしまえばカンタンなものです。

さて

と入力すると画面は

というふうに

ではありませんか！

前ページの画面と比べてみてください。

いやぁ、ビックリしま……せんでしたか。ハァ。

4・7　画面設定内部サブルーチンあれこれ

えー、これで皆さんは、

画面の文字数を設定する内部ルーチン

をマスターしたわけです。

内部ルーチンを活用する――といっても意外とたいしたことないですね。

『他に画面の設定する内部ルーチンはないのか』

その質問をワタクシは待っておりました。

画面の設定を行う内部ルーチンは、他にも数多くあります。

例えば

●ファンクションキーの有　無

●カラー・白黒モードの設定

■画面の消去

■画面のスクロール

等々――。

実は、コレでもほんの一部なんですね。

これだけ多くの内部ルーチンを前章の――画面の文字数を設定する内部ルーチン――のようにくわしく説明するとしたならば、本書は、全400ページという巨編になってしまうでしょう。

といっても、1つ、2つを説明してもあまり意味がないし――。

そこで！

この章では、これらの

画面設定内部ルーチン

の中から重要なものをピックアップし、それらの内部ルーチンを簡単に紹介することにいたしました。

この章を活用するか、しないか、それは

あなた自身が選ぶことです！

【画面のモードの設定ルーチン】

前章で、皆さんがマスターした―画面の文字数を設定する内部ルーチン――は、

6F63H番地の内部ルーチン

を使っていましたね。

実は、この

画面の文字数を設定する内部ルーチン

は。本当は

画面のモード設定を行う内部ルーチン

だったのです。

ですから、画面の文字数を設定する以外にも

●スクロール幅を変えたり

●カラー／白黒モードの設定を行ったり

●ファンクション・キーの有無

●ヌル・キャラクタの設定

を行うことができます。

今までは、この内部ルーチンの力を最大には活用していなかったんです。

では、この内部ルーチンの概要をもう一度見てみることにいたしましょう。

6F6BH：画面のモード設定を行う

6F6BH

画面のモード設定を行う

A、F、B、C、D、E

B、C、その他ワークエリア

Bレジスタに画面の桁数を入れ、Cレジスタに行数を入れて6F6BH番地を呼ぶことによって画面の文字数の設定を行う。

この時、同時にワークエリアによって他の画面モードも設定する。

ここで

注目！

してもらいたいのは、最後の

この時、同時にワークエリアによって

他の画面のモードも設定する

というところです。

えー

ワークエリア

というのは

N88－BASICが使用するメモリ領域

のことで、BASICが使用する様々な数値、データ等が保存されているのです。

たとえば、ワークエリアには

変数の値

画面のスクロール幅

ファンクション・キーの内容

カラー・コード

ファンクション・キー[illegible]示の有 [illegible]

等の値がデータとして保存されているわけなのです。

ですから

というのですから、ワークエリアに保存されているデータの値を変えてから、この――画面のモード設定を行う内部ルーチン――をコールすれば

変えたワークエリアの値により

[illegible]由に画面のモードを設定できる

わけです。

この

画面のモード設定を行う内部ルーチン

に関係する

は、次の通りです。

E6B2H	スクロール開始行(01H～19H)
E6B3H	スクロール終了行(01H～19H)
E6B4H	アトリビュート・コード
E6B5H	ヌル・キャラクタ
E6B6H	コントロール・コード・スイッチ
E6B8H	ファンクション・キー表示の有／無
E6B9H	カラー／白黒

これらのワークエリアを変更し、画面のモード設定を行う内部ルーチンをコールするだけで、たやすく画面のモードを設定することができます。

それでは、これらのワークエリアを変更して

ではありませんか。

【スクロール幅を設定】

それでは、まず

スクロール幅の設定

を行ってみることにいたしましょう。

スクロール幅のワークエリア

は、次の２つです。

E6B2H	スクロール開始行(01H～19H)
E6B3H	スクロール終了行(01H～19H)

この

スクロール開始行

スクロール終了行

のワークエリア

設定する行＋１

の値からなる１バイトとなっています。

サンプル スクロール幅を10行から20行とします。

```
アドレス マシン語
                    ; ****************************
                    ; * 6F6BH:CRT ヲ セッテイ スル      *
                    ; *        for PC-8801          *
                    ; *             by K.ツカモト *
                    ; ****************************
                    ;
                            ORG   0B900H
                    ;
B900  0650                  LD    B,80
B902  0E19                  LD    C,25
                    ;
B904  21B2E6                LD    HL,0E6B2H
B907  360B                  LD    (HL),10+1
B909  23                    INC   HL
B90A  3615                  LD    (HL),20+1
                    ;
B90C  CD6B6F                CALL  6F6BH
                    ;
B90F  3E04                  LD    A,4
B911  D331                  OUT   (31H),A
B913  CD4760                CALL  6047H
```

80×25に設定

10〜20にスクロール幅を設定

【アトリビュート・コード】

等というと

難かしそうだなあ

とお考えになるかもしれませんね。

しかし、この

アトリビュート・コード

なる難解な単語は、実は

のことなのです。

ですから、このアトリビュート・コードのワーク・エリアを変えることにより

キャラクタのカラーを
変えることができる

のです。

E6B4H	アトリビュート・コード

ワーク・エリアに書き込み指定する

は次の通りです。

カラー	キャラクタ・モード	グラフィック・モード
黒(0)	08H	18H
青(1)	28H	38H
赤(2)	48H	58H
紫(3)	68H	78H
緑(4)	88H	98H
シアン(5)	A8H	B8H
黄(6)	C8H	D8H
白(7)	E8H	F8H

サンプル カラーをシアン（水色）に設定する。

```
                ; ****************************
                ; * 6F6BH:CRT ヲ セッテイ スル      *
                ; *          for PC-8801       *
                ; *             by K.ツカモト  *
                ; ****************************
                ;
                        ORG   0B900H
                ;
B900 0628               LD    B,40
B902 0E19               LD    C,25
                ;
B904 21B4E6             LD    HL,0E6B4H
B907 36A8               LD    (HL),0A8H
                ;
B909 CD6B6F             CALL  6F6BH
                ;
B90C 3E04               LD    A,4
B90E D331               OUT   (31H),A
B910 CD4760             CALL  6047H
```

【ヌル・キャラクタの[illegible]】

またしても

[illegible]

なる、訳のわからない単語が出現してきましたね。

この

[illegible]

なる単語は、いったいどんな意味を示すのでしょうか？

ヌル・キャラクタは、N－BASICでは使われていましたが
N88－BASICでは、ほとんど使用されていないのです。

そこで、PC－8801を

N－BASICモード

にして

ヌル・キャラクタ

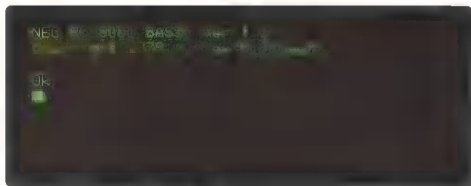
なるものを皆さんにお見せいたしましょう。

ではでは

```
[illegible]
```

と実行して、N－BASICモードにしてください。

おっと！　ディスクを使用している方は、ディスクのふたを開けて
おいてください。

すると、画面には

と表示され

になりました。

さて、N－BASICモードになったので

```
COLOR,236
```

とキー・インしてみてください。

この命令を実行すると

ヌル・キャラクタが
〈■〉に設定される

のです。

これで

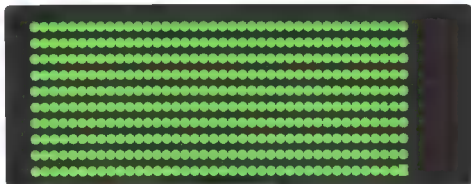
ヌル・キャラクタ

という、訳のわからないものが

わけです。

なぜ、〈●〉に設定されたんだ　－とお考えの方はリファレンス・マニュアルを読むことをお勧めいたします。

このような状態で

と画面は、次のページのようになります。

ナ、ナ、ナーント！

画面じゅうが〈■〉で埋まってしまったでは、ありませんか!!

ナゼ、このようなことが起きるのでしょうか？

実は

> **ヌル・キャラクタは**
> **画面クリアを行う時に埋めるキャラクタのこと**

だったのです。ジャジャーン！

だから

ヌル・キャラクタを
〈●〉に設定する

画面クリアした時
画面じゅうが〈■〉で埋まる

わけです。

が、しかし、コレは

N－BASICモードの場合だけ

でN88－BASICでは使われていません。

使われていないどころか

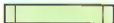
ヌル・キャラクタを設定する命令すらない

のです。

ところが、おもしろいことにマシン語で次のサンプル・プログラムのように

無理矢理にヌル・キャラクタを
〈■〉に設定する

と CTRL ＋ E や CTRL ＋ H を押した時に〈■〉が出現してしまいます。ヒエー!!

```
                        ; *************************
                        ; * 6F6BH:CRT ヲ セッテイ スル      *
                        ; *         for PC-8801      *
                        ; *               by K.ツカモト *
                        ; *************************
                        ;
                                ORG   0B900H
                        ;
B900 0628                       LD    B,40
B902 0E19                       LD    C,25
                        ;
B904 21B5E6                     LD    HL,0E6B5H
B907 36EC                       LD    (HL),0ECH
                        ;
B909 CD6B6F                     CALL  6F6BH
                        ;
B90C 3E04                       LD    A,4
B90E D331                       OUT   (31H),A
B910 CD4760                     CALL  6047H
```

4·8 PRINTするにはどうするか

皆さん、ご存じの通り

マシン語にはＰＲＩＮＴ命令がない

のです。

BASIC では

```
1000 FOR I=1 TO 5
1010   READ D$
1020     PRINT D$
1030 NEXT
1040 '
1100 DATA "
1110 DATA "
1120 DATA "
1130 DATA "
1140 DATA "
```

とすれば、カンタンに

というふうに、ＰＲＩＮＴ命令を使って画面に表示することができますね。

ところが！

マシン語にはＰＲＩＮＴ命令のようなものはないのですから、このようなキャラクタを表示しようにもできないのです。

どうしたらいいのでしょうか？

マシン語の高速性を利用して
高速データ処理を行っても
画面に表示できない

のではしかたありません。

この難問も内部ルーチンを利用すれば

簡単に解決でき■

のです。

【マシン語でもPRINTできる】

『どうやってや■んだ！』

マァマァ、解説はあとまわしにして、まずは

マシン語でメッセージを表示するプログラム

を見てもらうことにいたしましょう。

```
アドレス  マシン語
                    ; ****************************
                    ; * 5550H:CRT ニ メッセージ ヲ ダス ■
                    ; *            for PC-8801      *
                    ; *                by K.ツカモト   *   ←注釈
                    ; ****************************
                    ;
                           ORG   0B900H  ←── 900Hから作る
                    ;
B900 3E00                  LD    A,0  ←── Aに0を代入
B902 324CE6                LD    (0E64CH),A  ←
B905 2112B9                LD    HL,MSGDAT  ←── HLをMSGDATを代入
                    ;
B908 CD5055                CALL  5550H  ←── 内部ルーチンをコール
                    ;
B90B 3E04                  LD    A,4
B90D D331                  OUT   (31H),A
B90F CD4760                CALL  6047H      モニタに戻る
                    ;
B912 87878787       MSGDAT:DEFB '■■■ ■■■ ■■■',0DH,0AH
B916 20E48787
B91A E520E487
B91E 87E50D0A
B922 20202087              DEFB '   ■■ ■■ ■',0DH,0AH
B926 20872020
B92A 87208720
B92E 20870D0A
B932 87878787              DEFB '■■■  ■  ■■ ■',0DH,0AH
B936 20208787
B93A 20208787
B93E 20870D0A
B942 87202020              DEFB '■   ■ ■■ ■',0DH,0AH
B946 20872020
B94A 87208720
B94E 20870D0A
B952 87878787              DEFB '■■■ ■■■ ■■■',0DH,0AH
B956 20E68787
B95A E720E687
B95E 87E70D0A
B962 00                    DEFB 00H
                    ;
```

「見るだけじゃ面白くないゾ！」

ごもっともです。

プログラムを見るだけでは面白くありませんから、ココは、やはりプログラムを走らせてみるべきでしょう。えー、

アセンブラを[illegible]持ちの方

は、次のようにソース・リストを打ち込んでください。

まだ

アセンブラを■持ちでない方

は、次のようにキーインしてください。

```
mon

[illegible]
B900 [illegible] FF-01 FF-32 [illegible] [illegible]
B905 [illegible] FF-12 FF-b9 FF-ed FF-50
B90A FF-55 FF-3e FF-04 FF-d3 FF-31
B90F [illegible] FF-47 FF-60 FF-87 FF-87
B914 FF-87 FF-87 FF-20 FF-e4 FF-87
B919 FF-87 FF-e5 FF-20 FF-e4 FF-87
B91E FF-87 FF-e5 FF-0d FF-0a FF-20
B923 FF-20 FF-20 FF-87 FF-20 FF-87
B928 FF-20 FF-20 FF-87 FF-20 FF-87
B92D FF-20 FF-20 FF-87 FF-0d FF-0a
B932 FF-87 FF-87 FF-87 FF-87 FF-20
B937 FF-20 FF-87 FF-87 FF-20 FF-20
B93C FF-87 FF-87 FF-20 FF-87 FF-0d
B941 FF-0a FF-87 FF-20 FF-20 FF-20
B946 FF-20 FF-87 FF-20 FF-20 FF-87
B94B FF-20 FF-87 FF-20 FF-20 FF-87
B950 FF-0d FF-0a FF-87 FF-87 FF-87
B955 FF-87 FF-20 FF-e6 FF-87 FF-87
B95A FF-e7 FF-20 FF-e6 FF-87 FF-87
B95F FF-e7 FF-0d FF-0a FF-00
hJ
```

さーてさてさて、これで皆さんはプログラムをキーインしたわけですね。

それでは、このプログラムを

実行

してみることにいたしましょう。

どのような結果になるのか、楽しみですね。

このプログラムを実行するには、

```
gb900
```

とキーインします。

すると、画面には

というように、なんと

が表示されたのです。

「280って何のことだ？」

いやいや「280」ではなくワタクシは

Z80

と表示したつもりなのですが、ハア。

どうです。このように

マシン語でPRINTする
ことができる

のです。ナント！

どうやって表示しているのでしょうか？

もちろん、コレも

内部ルーチンを使っている

のですが、どの内部ルーチンをどのように使っているのでしょうか？

この章では、もっとも利用度の高い

について説明していくつもりです。

4・9　1文字表示内部ルーチン

えー、画面表示内部ルーチンの代表的なものに

1文字表示内部ルーチン

という内部ルーチンがあります。

この1文字表示内部ルーチンというのは、BASICでいえば

```
PRINT CHR$(&HEC);
```

```
PRINT CHR$(&HEC);
●
Ok
■
```

といったようなもののことなのです。

おっと！　この

&HEC

というのは

16進数のＥＣＨ

のことで、BASICで16進数を表わす場合には、頭に

&H

をつけることになっています。

【1文字表示の概要】

それでは、まず、この1文字表示ルーチンの

概要

から——。

3E0DH：画面へ1文字表示を行う

アドレス	3E0DH
機　能	画面への1文字表示を行う
レジスタ	全て保存される
引　数	A
説　明	Aレジスタの値をキャラクタ・コードとし、画面のカーソル位置に表示する。 Aレジスタの値が20H未満の場合、コントロール・コードとして扱い指定の動作を行う。

これが、この――1文字表示内部ルーチン――です。

「おい、キャラクタ・コードって何だ？」

ごもっともです。キャラクタ・コードについては、まだ説明しておりませんでしたね。

それでは、まず、この

キャラクタ・コード

から説明することにいたしましょう。

我々が通常使っている

Ra@ス$' r♠bタ

というようなキャラクタには

1つ1つに決まった値が対応している

のです。

この値を

キャラクタ・コード

というわけなのです。

これらのキャラクタとキャラクタ・コードとの対応表を

キャラクタ・コード表

といい、これがそのキャラクタ・コード表です。

この表を見れば

■ ————→ ECH
A ————→ 41H
タ ————→ C0H

というように、それぞれのキャラクタに対応したキャラクタ・コードを調べることができるわけです。

しかし！　しかし、いちいち調べるのは、めんどくさいのです。

そこで、アセンブラでは、

'●' ————→ ECH
'[illegible]' ————→ E4H,E5H
'A' ————→ 41H

というように、

'（アポストロフィ）で囲んだキャラクタを
自動的にキャラクタ・コードに変換する

ようになっています。

【神秘なるコントロール・コード】

さて、先ほどのキャラクタ・コード表の中には

コントロール・コード

というものがありましたね。

はたして、このコントロール・コードなるものはいったい何者なのでしょうか？

とにかく、実際に使ってみればわかることですから

実験

してみることにいたしましょう。まず、

0CH

のコントロール・コードから実験してみましょう。

それでは

```
PRINT CHR$(&H0C);
```

とキーインします。

もしも、これが普通のキャラクタであれば、当然

$^{C}{}_{L}$

といったキャラクタが表示されますね。

ところが。ところがなのです。

ナント、画面には、そのようなキャラクタは表示されず、

というように

のです。

　なぜ、このようなことが起きたのでしょうか？

　実は、コントロール・コードには

特別な意味がある

からなのです。

　コントロール・コードには、主に

コード			意　味
07H	B_L	BELL	ブザーを鳴らす
0AH	L_F	LINE FEED	1行改行
0BH	H_M	HOME	カーソルをホーム・ポジションへ移す
0CH	C_L	CLEAR	画面をクリアした後ホーム・ポジションへ
0DH	C_R	RETURN	カーソルを次の行の先頭に移す

といったものが用意されております。

【スペードはインベーダか？】

さて、キャラクタ・コードについて理解したのですから、今度はこの
——１文字表示内部ルーチン——を

活用

してみることにいたしましょう。

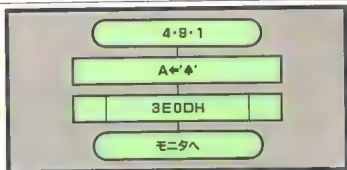
まずは画面に、

♠（スペード）

を表示するプログラムから——。

```
アドレス マシン語
                  ;  ****************************
                  ;  * 3E0DH:CRT ニ <♠> ヲ ヒョウジ *
                  ;  *          for PC-8801        *
                  ;  *               by K.ツカモト *    注釈
                  ;  ****************************
                  ;
                         ORG   0B900H   ←—— B900Hから作る
                  ;
B900  3EE8               LD    A,'♠'    ← Aに♠のコードを代入
                  ;
B902  CD0D3E             CALL  3E0DH    ← 内部ルーチンをコール
                  ;
B905  3E04               LD    A,4
B907  D331               OUT   (31H),A
B909  CD4760             CALL  6047H       モニタへ戻る
```

通常の処理

サブルーチン・コール

皆さん、ご存じの通り

LD　　A,♠

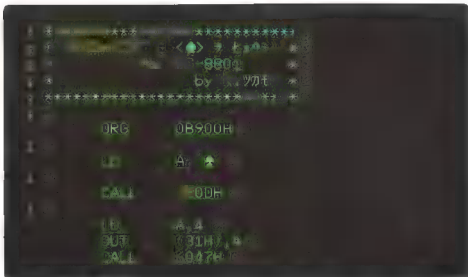
では、

Aレジスタに E 8 H を代入させる

ということを行っているわけですね。

それでは、このプログラムを打ち込むことにいたしましょう。
まず、

アセンブラをお持ちの方

は

とキーインします。
また、まだ

アセンブラをお持ちでない方

は、

とキーインします。

【スペード出現！】

これで皆さんは、全員

プログラムを入力し終えた

わけですね。
それでは、このプログラムを

実行

することにいたしましょう。

このマシン語プログラムはＤ０００Ｈ番地からプログラムが作られているのですから、

```
gb900
```

とキーインします。

すると、画面には

というように

が表示されましたね。

【ちょっとメッセージを】

このように

[illegible]

[illegible]

1 文 [illegible] 表示内部ルーチンをコー [illegible]

と、そのキャラクタを表示することができるわけです。

「1文字しか表示できないんじゃあ

使いもんになんないゾ！」

ごもっとも、ごもっとも。

しかし――チリも積れば山となる――というではありませんか。

1文字しか表示できなくても

10回繰り返せば

10文字表示できる

のです。

今度は、この――1文字表示内部ルーチン――を使って

メッセージ

を画面に表示することにいたしましょう。

「何が表示されるんだ？」

マァ、それはプログラムを

走らせてからの楽しみ

にとっておくことにいたしましょう。

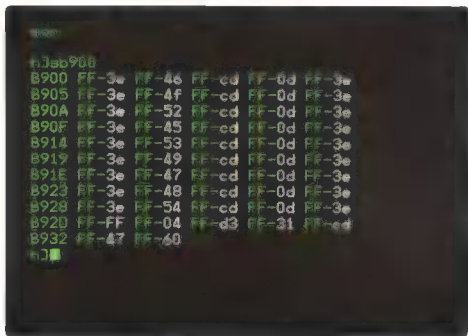
```
                    ; ****************************
                    ; * 3E0DH:CRT ﾒｯｾｰｼﾞ ｦ ﾋｮｳｼﾞ *
                    ; *           for PC-8801     *
                    ; *              by K.ﾂｶﾓﾄ    *
                    ; ****************************
                    ;
                           ORG   0B900H
                    ;
B900 3E46                  LD    A,'F'
B902 CD0D3E                CALL  3E0DH
                    ;
B905 3E4F                  LD    A,'O'
B907 CD0D3E                CALL  3E0DH
                    ;
B90A 3E52                  LD    A,'R'
B90C CD0D3E                CALL  3E0DH
                    ;
B90F 3E45                  LD    A,'E'
B911 CD0D3E                CALL  3E0DH
                    ;
B914 3E53                  LD    A,'S'
B916 CD0D3E                CALL  3E0DH
                    ;
B919 3E49                  LD    A,'I'
B91B CD0D3E                CALL  3E0DH
                    ;
B91E 3E47                  LD    A,'G'
B920 CD0D3E                CALL  3E0DH
                    ;
B923 3E48                  LD    A,'H'
B925 CD0D3E                CALL  3E0DH
                    ;
B928 3E54                  LD    A,'T'
B92A CD0D3E                CALL  3E0DH
                    ;
B92D 3E04                  LD    A,4
B92F D331                  OUT   (31H),A
B931 CD4760                CALL  6047H
```

4·9·2
A←'F'
3E0DH
A←'O'
3E0DH
A←'R'
3E0DH
A←'E'
3E0DH
A←'S'
3E0DH
A←'I'
3E0DH
A←'G'
3E0DH
A←'H'
3E0DH
A←'T'
3E0DH
モニタへ

また、いまだに

アセンブラを■持ちでない方

は、次のようにキーインしてください。

```
h]sb900
B900 FF-3e FF-46 FF-cd FF-0d FF-3e
B905 FF-3e FF-4f FF-cd FF-0d FF-3e
B90A FF-3e FF-52 FF-cd FF-0d FF-3e
B90F FF-3e FF-45 FF-cd FF-0d FF-3e
B914 FF-3e FF-53 FF-cd FF-0d FF-3e
B919 FF-3e FF-49 FF-cd FF-0d FF-3e
B91E FF-3e FF-47 FF-cd FF-0d FF-3e
B923 FF-3e FF-48 FF-cd FF-0d FF-3e
B928 FF-3e FF-54 FF-cd FF-0d FF-3e
B92D FF-FF FF-04 FF-d3 FF-31 FF-cd
B932 FF-47 FF-60
h]■
```

【おおっ!】

さて、

プログラムを入力し終った

わけですから、入力したプログラムを

実行

してみることにいたしましょう。

このプログラムを実行するには

```
gb900
```

とキーインします。

すると、画面には

とメッセージが表示されるではありませんか！

皆さんは、ついに

マシン語でメッセージを表示することに成功

したのです。

ここで、もう1度、このプログラムを見ることにいたしましょう。

```
                   ; *****************************
                   ; * 3E0DH:CRT ﾒｯｾｰｼﾞ ｦ ﾋｮｳｼﾞ  *
                   ; *          for PC-8801      *
                   ; *               by K.ﾂｶﾓﾄ   *
                   ; *****************************
                   ;
                           ORG    0B900H
                   ;
B900 3E46                  LD     A,'F'
B902 CD0D3E                CALL   3E0DH
                   ;
B905 3E4F                  LD     A,'O'
B907 CD0D3E                CALL   3E0DH
                   ;
B90A 3E52                  LD     A,'R'
B90C CD0D3E                CALL   3E0DH
                   ;
B90F 3E45                  LD     A,'E'
B911 CD0D3E                CALL   3E0DH
                   ;
B914 3E53                  LD     A,'S'
B916 CD0D3E                CALL   3E0DH
                   ;
B919 3E49                  LD     A,'I'
B91B CD0D3E                CALL   3E0DH
                   ;
B91E 3E47                  LD     A,'G'
B920 CD0D3E                CALL   3E0DH
                   ;
B923 3E48                  LD     A,'H'
B925 CD0D3E                CALL   3E0DH
                   ;
B928 3E54                  LD     A,'T'
B92A CD0D3E                CALL   3E0DH
                   ;
B92D 3E04                  LD     A,4
B92F D331                  OUT    (31H),A
B931 CD4760                CALL   6047H
```

どうです？　ただ単に

と表示するプログラムの割に

かなり長いプログラム

ですね。

これでは

ちょっとした説明を表示するプログラム

をマシン語で作ろうとしても

恐ろしく長いプログラム

になってしまいますね。

どうにかならないのでしょうか？

【こうすればいいのです】

どうにかなってしまうんですね、コレが。

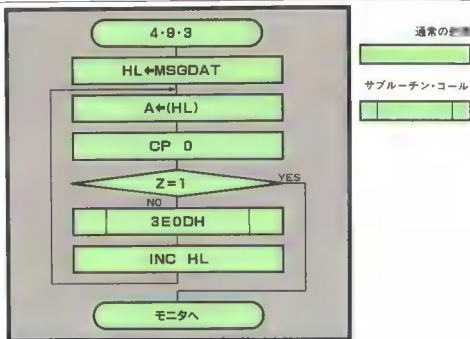
前にも説明しましたように

マシン語の作り方によって
スピード、プログラム長さが
まったく違う

のです。

とにかく、どのくらい違うのか見てもらうことにいたしましょう。

```
                  ; ****************************
                  ; * 3E0DH:CRT ﾒｯｾｰｼﾞ ｦ ﾋｮｳｼﾞ2 *
                  ; *           for PC-8801    *
                  ; *             by K.ﾂｶﾓﾄ    *
                  ; ****************************
                  ;
                          ORG   0B900H
                  ;
B900  2117B9              LD    HL,MSGDAT
                  ;
B903  7E          LOOP1:  LD    A,(HL)
                  ;
B904  FE00                CP    0
B906  CA10B9              JP    Z,EXIT
                  ;
```

```
B909 CD0D3E              CALL  3E0DH
                ;
B90C 23                  INC   HL
B90D C303B9              JP    LOOP1
                ;
B910 3E04       EXIT:    LD    A,4
B912 D331                OUT   (31H),A
B914 CD4760              CALL  6047H
                ;
B917 464F5245   MSGDAT:DEFB  'FORESIGHT'
B91B 53494748
B91F 54
B920 00                  DEFB  0
```

どうです？

かなりわかりやすくなった

のではないでしょうか。

しかも、このプログラムの場合、表示したいメッセージを変えたいと思ったならば、

```
MSGDAT:DEFB 'FORESIGHT'
```

を

```
MSGDAT:DEFB 'K.ツカモト ハ エライ!!'
```

というふうに差し替えるだけで、メッセージの文字数にカンケイなく変えることができるのです。

【動くも八掛、動かぬも八掛】

このような優れた構造のプログラムであっても

動かなければ、絵に描いたモチ

ですね。

本当に動くのか？

実際に実験してみることにいたしましょう。

アセンブラをお持ちの方

は、次のように入力してください。

とキーインします。

また、まだ

アセンブラをお持ちでない方

は、次のように入力してください。

```
]sb900
B900 FF-21 FF-17 FF-c9 FF-7e FF-fe
B905 FF-00 FF-ca FF-10 FF-b9 FF-cd
B90A FF-0d FF-3e FF-23 FF-c3 FF-03
B90F FF-b9 FF-3e FF-04 FF-d3 FF-31
B914 FF-cd FF-47 FF-60 FF-46 FF-4f
B919 FF-52 FF-45 FF-53 FF-49 FF-47
B91E FF-48 FF-54 FF-00
]
```

このように入力し終りましたら

```
gb900
```

としてプログラムを実行します。

すると画面には、

というようにメッセージが表示されました。
どうです？
ちゃんと動きましたね。

【なぜ動くのか？】

「動くのはわかったから
どうして動くのか説明してくれ」

わかりました。
それではこのプログラムが
どのように動いているか
を説明することにいたしましょう。

```
                    ; ****************************
                    ; * 3E0DH:CRT ﾒｯｾｰｼﾞ ｦ ﾋｮｳｼﾞ2 *
                    ; *          for PC-8801      *
                    ; *              by K.ﾂｶﾓﾄ    *
                    ; ****************************
                    ;
                            ORG   0B900H     ←B900Hから作る
                    ;
B900  2117B9                LD    HL,MSGDAT  ←を代入
                    ;
B903  7E            LOOP1:  LD    A,(HL)     ←AにHLの指すアドレスの内容を代入
                    ;
B904  FE00                  CP    0          ←Aと0を比較
B906  CA10B9                JP    Z,EXIT     ←Z=1ならEXITへ
                    ;
B909  CD0D3E                CALL  3E0DH      ←内部ルーチンをコール
                    ;
B90C  23                    INC   HL         ←HL=HL+1
B90D  C303B9                JP    LOOP1      ←LOOP1へ
                    ;
B910  3E04          EXIT:   LD    A,4
B912  D331                  OUT   (31H),A
B914  CD4760                CALL  6047H      ←モニタへ戻る
                    ;
B917  464F5245      MSGDAT:DEFB 'FORESIGHT'
B91B  53494748
B91F  54
B920  00                    DEFB  0
```

この命令を行うと、次に

```
LD        A,(HL)
```

という命令を行い

アキュームレータにHLレジスタの指す
アドレスの内容を移す

わけです。

こういうと、何がなんだかわかりませんが、ようするに

アキュームレータにメッセージ・テーブルの
先頭のキャラクタ(キャラクタ・コード)が入る

わけなのです。

こうして、アキュームレータにメッセージのキャラクタ・コードを代入したら、次は

```
CP        0
```

を行って

アキュームレータの内容と
0を比較

します。

この時、Z(ゼロ)フラグは

●アキュームレータの内容が0の場合
　→　Z＝1
●アキュームレータの内容が0でない場合
　→　Z＝0

というように変化します。

ここでは

アキュームレータが0かどうか

を判断しているのです。

そして、このフラグの変化を見て

```
JP        Z,EXIT
```

では、

●Z＝1ならば
モニタへジャンプ

しているわけです。

ということは

アキュームレータの値が0であれば
モニタに戻る

ということですね。

アキュームレータが0の時――そうです。メッセージを表示し終った時に、モニタに戻るということなのです。

アキュームレータに表示したいキャラクタのキャラクタ・コードが入っているわけですから

```
CALL      3E0DH
```

というように

1文字表示内部ルーチンをコール

すると、

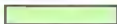
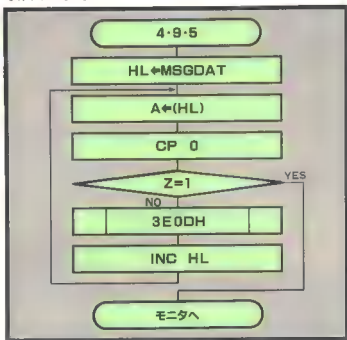
アキュームレータに入っている
キャラクタが表示される

わけです。

こうやって、1文字表示し終ったならば、次の文字を表示しなければなりませんね。

そこで

```
INC      HL
```

を行うわけです。

4・10 マシン語で文字列をPRINT！

前章で使った画面表示内部ルーチンは

1文字ずつしか表示できない

のでしたね。

1文字表示内部ルーチンの概要は次の通りです。

3E0DH：画面に1文字表示を行う

アドレス	3E0DH
機能	画面に1文字表示を行う。
レジスタ	全て保存
引数	A
説明	アキュームレータの内容をキャラクタ・コードとしたキャラクタを画面上のカーソルの位置に表示する。 アキュームレータの値がコントロール・コードを指す場合、指定された動作を行う。

この1文字表示内部ルーチンは非常に便利で有効な内部ルーチンです。

しかし

1文字ずつしか表示できない

のが玉にキズ。

なんとか、もっとカンタンに表示できる命令はないものでしょうか。

【文字列で表示】

ところが、ところがあるのです。

コレだから。内部ルーチンを活用しないとソンなのです。

まずは、その便利な内部ルーチンの

[illegible]

から――。

5550H：各周辺機器への文字列出力を行う

アドレス	5550H
機能	各周辺機器への文字列出力を行う。
レジスタ	A、F、B、C、D、E、H、L
引数	HL、E64CH

説　明　ＨＬレジスタによって指定されたアドレスから０が入っているアドレスのデータをキャラクタ・コードとし、任意の周辺機器に出力する。
周辺機器の指定はＥ６４ＣＨ番地の内容によって決められ、Ｅ６４ＣＨ番地の値が００Ｈであれば画面へ、０１Ｈ～７ＦＨであればプリンタへ。８０Ｈ～ＦＦＨであればカセットへ出力する。
ただし、データは255文字まで。

『どうも良くわからないのだが』

ごもっともです。

しかし、それほど気になさらなくても結構です。

一見、[illegible]に思えるこの——概要——も、文章は少々カタイようですが中身はたいしたことないのですから。

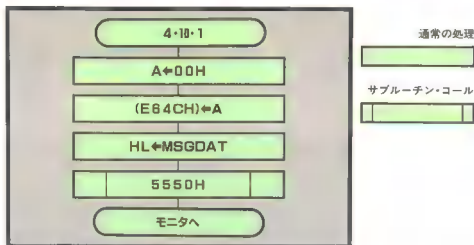
マア、とにかく、この——文字列出力内部ルーチン——とやらを

活用

してみることにいたしましょう。

```
                   ; ******************************
                   ; * 5550H:CRT ニ モジ゛レツ ヲ ヒョウジ゛ *
                   ; *          for PC-8801          *
                   ; *            by K.ツカモト      *
                   ; ******************************
                   ;
                           ORG   0B900H        ; B900Hから作る
                   ;
B900 3E00                  LD    A,00H
B902 324CE6                LD    (0E64CH),A    ; E64CHに0を代入
                   ;
B905 2112B9                LD    HL,MSGDAT     ; HLにMSGDATを代入
                   ;
B908 CD5055                CALL  5550H
                   ;
B90B 3E04          EXIT:   LD    A,4
B90D D331                  OUT   (31H),A
B90F CD4760                CALL  6047H         ; モニタへ戻く
                   ;
B912 CFBCDDBA      MSGDAT:DEFB ´マシンコ゛ カツヨウ ニュウモン!!´
B916 DE20B6C2
B91A D6B320C6
B91E ADB3D3DD
B922 2121
B924 00                    DEFB  0
```

えー、このプログラムは、画面に

[illegible]

というメッセージを表示するプログラムのはずです。

プログラムのはず——実はワタクシ、このプログラムを走らせていないのです。

ですから

本当に動くかわからない

わけなのです。

「いいかげんやなァ」

そうではないのです。皆さんと共に私自身プログラムを作ってこそ、皆さんにわかってもらえるのではないか——そう考えて、まだプログラムを走らせていないのです。

サァ、はたして本当にプログラムが走って

[illegible]コ゛　クツヨウ　ニュウモン!!

というようなメッセージが表示されるでしょうか？

途中で

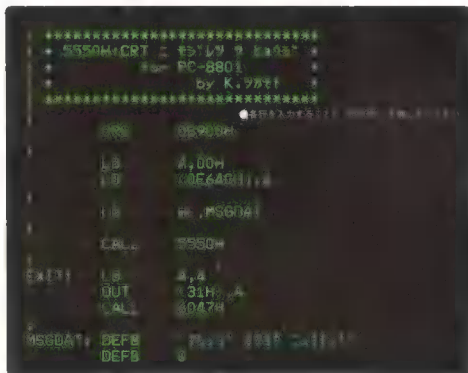
[illegible]

するようなことはないでしょうか？

私は、かなり心配なのです。

なぜなら、マシン語のプログラムは

思ってもみないマチガイがある

ことが多いからなのです。

```
B90F FF-cd [illegible]
B914 FF-dd [illegible]
B919 FF-c2 [illegible] FF-c6
B91E FF-ad [illegible]
B923 FF-21 FF-00
h]■
```

とキーインします。

これで、皆さんは

プログラムを打ち込み終った

わけですね。

それでは、プログラムを走らせてみることにしましょう。

プログラムを実行するには

```
gb900
```

とキーインします。

ややっ！

というようにメッセージが表示されたようです。

いやはや、良かった良かった。

【なぜ動くのか】

サテ

プログラムが動くことを確認した

のですから、ここで

[illegible]

を確認しておくことにいたしましょう。

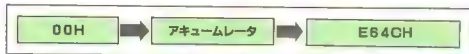
まあ、このようなことを行う必要はない——と思いますが。

まず、プログラムの

```
LD        A,00H       ←――― Aレジスタに00Hを代入
LD        (0E64CH),A  ←――― E64CH番地にAレジスタの内容を代入
```

では

E64CH番地に00Hを代入

しています。

なぜ、E64CH番地に00Hを代入しなければならないのでしょうか？

先はどの

内部ルーチンの概要

を思い出してください。

内部ルーチンの概要には

> **説　明**　HLレジスタによって指定されたアドレスから0が入っているアドレスのデータをキャラクタ・コードとし、任意の周辺機器に出力する。
> 周辺機器の指定はE64CH番地の内容によって決められ、E64CH番地の値が00Hであれば画面へ、01H～7FHであればプリンタへ、80H～FFHであればカセットへ出力する。
> ただし、データは255文字まで。

と書かれていましたね。

E64CH番地	周　辺　機　器
00H	C R T
01H～7FH	プ リ ン タ
80H～FFH	レ コ ー ダ

ですから

出力する周辺機器を画面にする

ために

E64CH番地に00Hを代入している

わけです。

さて、次は

```
LD        HL,MSGDAT                MSGDATの値を代入
```

ですね。

この命令では

HLレジスタにラベルMSGDAT

の値を代入

しています。

ラベル

MSGDAT

の値は

```
; *****************************
; * 5550H:CRT ニ モシ゛レツ ヲ ヒョウシ゛ *
; *          for PC-8801          *
; *              by K.ツカモト    *
; *****************************
;
        ORG     0B900H        B900Hから
;
        LD      A,00H
        LD      (0E64CH),A     E64CH
;
        LD      HL,MSGDAT
;
        CALL    5550H
;
EXIT:   LD      A,4
        OUT     (31H),A
        CALL    6047H
;
MSGDAT: DEFB    'マシンコ゛ カツヨウ ニュウモン!!'
        DEFB    0
```

を見ておわかりの通り

表示するデータの先頭を表している

わけです。

このように

引数の設定

を行ってから

```
CALL    5550H
```

というように

文字列出力内部ルーチンをコール

しているわけです。

[もっと面白く]

これで

文字列出力内部ルーチンの使い方

について皆さんにおわかりいただけたと思います。

しかし、コレだけでは

FORESIGHT

というようなメッセージしか表示できませんね。

やはり、わたくしとしては

というような

キャラクタ

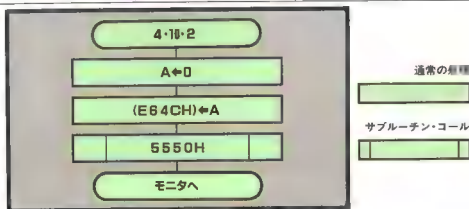
をヘコヘコと表示したいものです。

どうにかならないものでしょうか？

あ！　そういえば前に

と画面に表示するプログラムがありましたね。

これに**秘密が隠れている**ようです。

```
                           ; ******************************
                           ; * 5550H:CRT ニ メッセージ ヲ ダス *
                           ; *          for PC-8801          *
                           ; *               by K.ツカモト   *
                           ; ******************************
                           ;
                                   ORG   0B900H
                           ;
B900 3E00                          LD    A,0
B902 324CE6                        LD    (0E64CH),A
B905 2112B9                        LD    HL,MSGDAT
                           ;
B908 CD5055                        CALL  5550H
                           ;
B90B 3E04                          LD    A,4
B90D D331                          OUT   (31H),A
B90F CD4760                        CALL  6047H
                           ;
B912 87878787              MSGDAT:DEFB  '■■■ ▲▲ ▲▲',0DH,0AH
B916 20E48787
B91A E520E487
B91E 87E50D0A
B922 20202087                      DEFB  '   ■ ■ ■ ■ ■',0DH,0AH
B926 20872020
B92A 87208720
B92E 20870D0A
B932 87878787                      DEFB  '■■■■ ■■ ■■ ■',0DH,0AH
B936 20208787
B93A 20208787
B93E 20870D0A
B942 87202020                      DEFB  '■   ■ ■ ■ ■',0DH,0AH
B946 20872020
B94A 87208720
B94E 20870D0A
B952 87878787                      DEFB  '■■■ ▼▼ ▼▼',0DH,0AH
B956 20E68787
B95A E720E687
B95E 87E70D0A
B962 00                            DEFB  00H
                           ;
```

プログラムでは、特に変ったことを行っている様子はありませんね。
それではどこが違うのでしょうか？

メッセージのデータ

ややっ！　メッセージのデータに、なにやら

　　　　00H,0AH

というようなものがあるではありませんか。

ナ、ナンダこれは!!

地球侵略をねらうインベーダか、エイリアンか、ゴジラか、クジラか！

いやいや、コレはコントロール・コードのようです。

コントロール・コード――前に出てきましたね。このコントロール・コードを画面に出力すると任意の働きをするというものです。

この

　　　　00H,0AH

というコントロール・コードを使うと、カーソルの位置を

　　　　[illegible]行の先頭にもって[illegible]

ことができるのです。

ですから

```
MSGDAT: DEFB    '[illegible]',0DH,0AH
        DEFB    '[illegible]',0DH,0AH
        DEFB    '[illegible]',0DH,0AH
        DEFB    '[illegible]',0DH,0AH
        DEFB    '[illegible]',0DH,0AH
        DEFB    00H
;
```

というデータを文字列出力内部ルーチンへ出力するだけで

と画面に表示できるわけです。

第5章

次のステップへ

【内部ルーチン】

0000H	ＰＣ-8801のリセットを行う
4B1AH	BASICに制御を移す
0257H	ＣＲＴに１文字表示を行う
8F6BH	画面のモード設定を行う
5550H	周辺機器への文字列出力
6047H	モニタに制御を移す

さあ、どうです？

マア，内部ルーチンを覚える必要はありませんが、命令や基礎用語を覚えてますか。確認しておきましょう。

【次のステップへ】

皆さんは、数多くの

内部ルーチン

を活用しましたね。

これらの内部ルーチンは、皆さんがこれからマシン語を使っていくうえで非常に役立つことでしょう。

しかしながら、このような有効な内部ルーチンは

まだまだ数多くある

のです。

これらの内部ルーチンを知るには、どうしたら良いのでしょうか？

また、マシン語には数々のテクニックがあるのですが、これらを知るには、どうしたら良いのでしょうか？

このような方のために

マシン語活用研究

を予定しております。

ご期待ください。

1984年３月１日

フォーサイト企画部

塚本浩二

第6章

μCOM-82インストラクション活用表

Z80 インストラクション・セット

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210	
8ビット・ロード命令	LD r, r'	r←r'	•	•	•	•	•	•	1	4	01 r r'	Ⓔ
	LD r, n	r←n	•	•	•	•	•	•	2	7	00 r 110 ← n →	[illegible]
	LD r, (HL)	r←(HL)	•	•	•	•	•	•	1	7	01 r 110	Ⓔ
	LD r, (IX+d)	r←(IX+d)	•	•	•	•	•	•	3	19	11 011 101 01 r 110 ← d →	Ⓔ
	LD r, (IY+d)	r←(IY+d)	•	•	•	•	•	•	3	19	11 111 101 01 r 110 ← d →	Ⓔ
	LD (HL), r	(HL)←r	•	•	•	•	•	•	1	7	01 110 r	Ⓔ
	LD (IX+d), r	(IX+d)←r	•	•	•	•	•	•	3	19	11 011 101 01 110 r ← d →	Ⓔ
	LD (IY+d), r	(IY+d)←r	•	•	•	•	•	•	3	[illegible]	11 111 101 01 110 r ← d →	[illegible]
	LD (HL), n	(HL)←n	•	•	•	•	•	•	2	10	00 110 110 ← n →	
	LD (IX+d), n	(IX+d)←n	•	•	•	•	•	•	4	19	11 011 101 [illegible] 110 110 ← d → ← n →	
	LD (IY+d), n	(IY+d)←n	•	•	•	•	•	•	4	19	11 111 101 00 110 110 ← d → ← [illegible] →	
	LD A, (BC)	A←(BC)	•	•	•	•	•	•	1	7	00 001 010	
	LD A, (DE)	A←(DE)	•	•	•	•	•	•	1	7	00 011 010	
	LD A, (nn)	A←(nn)	•	•	•	•	•	•	3	13	00 111 010 ← n → ← [illegible] →	
	LD (BC), A	(BC)←A	•	•	•	•	•	•	1	7	00 000 010	
	LD (DE), A	(DE)←A	•	•	•	•	•	•	1	7	00 010 010	
	LD (nn), A	(nn)←A	•	•	•	•	•	•	3	13	00 110 010 ← n → ← n →	
	LD A, I	A←I	↕	↕	0	IFF	0	•	[illegible]	9	11 101 101 01 010 111	
	LD A, R	A←R	↕	↕	0	IFF	0	•	2	9	11 101 101 01 011 111	
	LD I, A	I←A	•	•	•	•	•	•	2	9	11 101 101 01 000 111	
	LD R, A	R←A	•	•	•	•	•	•	2	9	11 101 101 01 001 111	
16ビット・ロード命令	LD dd, nn	dd←nn	•	•	•	•	•	•	3	10	00 dd0 001 ← n → ← n →	Ⓐ
	LD IX, nn	IX←nn	•	•	•	•	•	•	4	14	11 011 101 00 100 001 ← n → ← n →	
	LD IY, nn	IY←nn	•	•	•	•	•	•	4	14	11 111 101 00 100 001 ← n → ← n →	
	LD HL, (nn)	H←(nn+1) L←(nn)	•	•	•	•	•	•	3	16	00 101 010 ← n → ← n →	
	LD dd, (nn)	dd_H←(nn+1) dd_L←(nn)	•	•	•	•	•	•	4	20	11 101 101 01 dd1 011 ← n → ← n →	Ⓐ

命令群	ニーモニック	オペレーション	フラグ S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210	
16ビット・ロード命令	LD IX, (nn)	IX_H←(nn+1) IX_L←(nn)	·	·	·	·	·	·	4	20	11 011 101 [illegible] 101 010 ← n → ← n →	
	LD IY, (nn)	IY_H←(nn+1) IY_L←(nn)	·	·	·	·	·	·	4	20	11 111 101 00 101 010 ← n → ← n →	
	LD (nn), HL	(nn+1)←H (nn)←L	·	·	·	·	·	·	3	16	00 100 010 ← n → ← n →	
	LD (nn), dd	(nn+1)←dd_H (nn)←dd_L	·	·	·	·	·	·	4	20	11 101 101 01 dd0 011 ← n → ← n →	Ⓐ
	LD (nn), IX	(nn+1)←IX_H (nn)←IX_L	·	·	·	·	·	·	4	20	11 011 101 00 100 010 ← n → ← n →	
	LD (nn), IY	(nn+1)←IY_H (nn)←IY_L	·	·	·	·	·	·	4	20	11 111 101 00 100 010 ← n → ← n →	
	LD SP, HL	SP←HL	·	·	·	·	·	·	1	[illegible]	11 111 001	
	LD SP, IX	SP←IX	·	·	·	·	·	·	2	10	11 011 101 11 111 001	
	LD SP, IY	SP←IY	·	·	·	·	·	·	2	10	11 111 101 11 111 001	
	PUSH [illegible]	(SP−2)←qq_L (SP−1)←qq_H	·	·	·	·	·	·	1	11	11 qq0 101	Ⓑ
	PUSH IX	(SP−2)←IX_L (SP−1)←IX_H	·	·	·	·	·	·	2	15	11 011 101 11 [illegible] 101	
	PUSH IY	(SP−2)←IY_L (SP−1)←IY_H	·	·	·	·	·	·	2	15	11 111 101 11 100 101	
	POP qq	qq_H←(SP+1) qq_L←(SP)	·	·	·	·	·	·	1	10	11 qq0 001	Ⓑ
	POP IX	IX_H←(SP+1) IX_L←(SP)	·	·	·	·	·	·	2	14	11 011 101 11 100 001	
	POP IY	IY_H←(SP+1) IY_L←(SP)	·	·	·	·	·	·	2	14	11 111 101 11 100 001	
エクスチェンジ命令	EX DE, HL	DE↔HL	·	·	·	·	·	·	1	4	11 101 011	
	EX AF, AF'	AF↔AF'	·	·	·	·	·	·	1	4	00 001 [illegible]	
	EXX	BC↔BC' DE↔DE' HL↔HL'	·	·	·	·	·	·	1	4	11 011 001	
	EX (SP), HL	H↔(SP+1) L↔(SP)	·	·	·	·	·	·	1	[illegible]	11 100 011	
	EX (SP), IX	IX_H↔(SP+1) IX_L↔(SP)	·	·	·	·	·	·	2	23	11 011 101 11 100 011	
	EX (SP), IY	IY_H↔(SP+1) IY_L↔(SP)	·	·	·	·	·	·	2	23	11 111 101 11 [illegible] 011	
ブロック転送命令	LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	·	·	[illegible]	① ↕	0	·	2	[illegible]	11 101 101 10 100 000	
	[illegible]	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 until BC=0	·	·	0	0	0	·	2	21 if BC≠0 16 if BC=0	11 101 101 10 110 [illegible]	

命令群	ニーモニック	オペレーション	フラグ						バイト	ステート	OPコード	
			S	Z	H	P/V	N	C			76 543 210	
ブロック転送命令	LDD	(DE)←(HL) DE←DE－1 HL←HL－1 BC←BC－1	・	・	0	↕ ①	0	・	2	16	11 101 101 10 101 000	
	LDDR	(DE)←(HL) DE←DE－1 HL←HL－1 BC←BC－1 until BC＝0	・	・	0	0	0	・	2	21 if BC≠0 16 if BC＝0	11 101 101 10 111 000	
ブロックサーチ命令	CPI	A－(HL) HL←HL＋1 BC←BC－1	↕	↕ ②	↕	↕ ①	1	・	2	16	11 101 101 10 100 001	
	CPIR	A－(HL) HL←HL＋1 BC←BC－1 until A＝(HL) or BC＝0	↕	↕ ②	↕	↕ ①	1	・	2	21 if BC≠0 and A≠(HL) 16 if BC＝0 or A＝(HL)	11 101 101 10 110 001	
	CPD	A－(HL) HL←HL－1 BC←BC－1	↕	↕ ②	↕	↕ ①	1	・	2	16	11 101 101 10 101 001	
	CPDR	A－(HL) HL←HL－1 BC←BC－1 until A＝(HL) or BC＝0	↕	↕ ②	↕	↕ ①	1	・	2	21 if BC≠0 and A≠(HL) 16 if BC＝0 or A＝(HL)	11 101 101 10 111 001	
8ビット算術論理演算命令	ADD A, r	A←A＋r	↕	↕	↕	V	0	↕	1	4	10 000 r	Ⓔ
	ADD A, n	A←A＋n	↕	↕	↕	V	0	↕	2	7	11 000 110 ← n →	
	ADD A, (HL)	A←A＋(HL)	↕	↕	↕	V	0	↕	1	7	10 000 110	
	ADD A, (IX＋d)	A←A＋(IX＋d)	↕	↕	↕	V	0	↕	[illegible]	19	11 011 101 10 000 110 ← d →	
	ADD, A, (IY＋d)	A←A＋(IY＋d)	↕	↕	↕	V	0	↕	3	19	11 111 101 10 000 110 ← d →	
	ADC A, r	A←A＋r＋CY	↕	↕	↕	V	0	↕	1	[illegible]	10 001 r	Ⓔ
	ADC A, n	A←A＋n＋CY	↕	↕	↕	V	0	↕	2	7	11 001 110 ← n →	
	ADC A, (HL)	A←A＋(HL)＋CY	↕	↕	↕	V	0	↕	1	7	10 001 110	
	ADC A, (IX＋d)	A←A＋(IX＋d)＋CY	↕	↕	↕	V	0	↕	3	[illegible]	11 011 101 10 001 110 ← d →	
	ADC A, (IY＋d)	A←A＋(IY＋d)＋CY	↕	↕	↕	V	0	↕	3	19	11 111 101 10 001 110 ← d →	
	SUB r	A←A－r	↕	↕	↕	V	1	↕	1	[illegible]	10 010 r	Ⓔ
	SUB n	A←A－n	↕	↕	↕	V	1	↕	[illegible]	7	11 010 110 ← n →	
	SUB (HL)	A←A－(HL)	↕	↕	↕	V	1	↕	1	7	10 010 110	
	SUB (IX＋d)	A←A－(IX＋d)	↕	↕	↕	V	1	↕	3	19	11 011 101 10 010 110 ← d →	
	SUB (IY＋d)	A←A－(IY＋d)	↕	↕	↕	V	1	↕	3	19	11 111 101 10 010 110 ← d →	
	SBC A, r	A←A－r－CY	↕	↕	↕	V	1	↕	1	[illegible]	10 011 r	[illegible]
	SBC A, n	A←A－n－CY	↕	↕	↕	V	1	↕	2	7	11 011 110 ← n →	
	SBC A, (HL)	A←A－(HL)－CY	↕	↕	↕	V	1	↕	1	7	10 011 110	

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
8ビット算術論理演算命令	DEC (IY+d)	(IY+d)←(IY+d)−1	↕	↕	↕	V	1	•	3	23	11 111 101 00 110 101 ← d →
16ビット算術演算命令	ADD HL, ss	HL←HL+ss	•	•	×	•	0	↕	1	11	■ ss1 001 Ⓐ
	ADC HL, ss	HL←HL+ss+CY	↕	↕	×	V	0	↕	2	15	11 101 101 ■ ss1 010 Ⓐ
	SBC HL, ss	HL←HL−ss−CY	↕	↕	×	V	1	↕	■	15	11 101 101 01 ss0 010 Ⓐ
	ADD IX, pp	IX←IX+pp	•	•	×	•	0	↕	2	15	11 011 101 ■ pp1 001 Ⓒ
	ADD IY, rr	IY←IY+rr	•	•	×	•	0	↕	2	15	11 111 101 00 rr1 001 Ⓓ
	INC ■	ss←ss+1	•	•	•	•	•	•	1	6	■ ss0 011 Ⓐ
	INC IX	IX←IX+1	•	•	•	•	•	•	2	10	11 011 101 00 100 011
	INC IY	IY←IY+1	•	•	•	•	•	•	2	10	11 111 101 00 100 011
	DEC ss	ss←ss−1	•	•	•	•	•	•	1	6	■ ss1 011 Ⓐ
	DEC IX	IX←IX−1	•	•	•	•	•	•	2	10	11 011 101 ■ 101 011
	DEC IY	IY←IY−1	•	•	•	•	•	•	2	10	11 111 101 ■ 101 011
アキュムレータ操作命令	DAA	Decimal adjust Acc	↕	↕	↕	P	•	↕	■	4	■ 100 111
	CPL	A←Ā	•	•	1	•	1	•	1	4	■ 101 111
	NEG	A←Ā+1	↕	↕	↕	V	1	↕	2	■	11 101 101 01 000 100
	CCF	CY←$\overline{CY}$	•	•	×	•	0	↕	1	4	00 111 111
	SCF	CY←1	•	•	0	•	0	1	1	4	00 110 111
CPUコントロール命令	NOP	No operation	•	•	•	•	•	•	1	4	00 ■ 000
	HALT	CPU halted	•	•	•	•	•	•	1	4	01 110 110
	DI	IFF←0	•	•	•	•	•	•	1	4	11 110 011
	EI	IFF←1	•	•	•	•	•	•	1	4	11 111 011
	IM 0	Set interrupt mode 0	•	•	•	•	•	•	2	8	11 101 101 01 000 110
	IM 1	Set interrupt mode 1	•	•	•	•	•	•	2	■	11 101 101 01 010 110
	IM 2	Set interrupt mode 2	•	•	•	•	•	•	2	8	11 101 101 01 011 110
ローテート・シフト命令	RLCA	CY ← [7 ← 0] (A)	•	•	0	•	0	↕	1	4	00 000 111
	RLA	[CY] ← [7 ← 0] (A)	•	•	0	•	0	↕	1	4	00 010 111
	RRCA	[7 → 0] → CY (A)	•	•	0	•	0	↕	1	4	00 001 111
	RRA	[7 → 0] → [CY] (A)	•	•	0	•	0	↕	1	4	00 011 111

命令群	ニーモニック	オペレーション	フラグ S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210	
ローテート・シフト命令	RLC r		↕	↕	0	P	0	↕	2	8	11 001 011 00 000 r	Ⓔ
	RLC (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 000 110	
	RLC (IX+d)	CY ← 7 ← 0 (7→0 循環) r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 000 110	
	RLC (IY+d)		↕	↕	0	P	[illegible]	↕	4	23	11 111 101 11 001 011 ← d → 00 000 110	
	RL r		↕	↕	0	P	0	↕	2	8	11 001 011 00 010 r	Ⓔ
	RL (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 010 110	
	RL (IX+d)	CY ← 7 ← 0 ← (CY) r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	[illegible]	23	11 011 101 11 001 011 ← d → 00 010 110	
	RL (IY+d)		↕	↕	0	P	0	↕	[illegible]	23	11 111 101 11 001 011 ← d → 00 010 110	
	RRC r		↕	↕	0	P	0	↕	[illegible]	[illegible]	11 001 011 00 001 r	Ⓔ
	RRC (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 001 110	
	RRC (IX+d)	(0→7 循環) 7 → 0 → CY r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 001 110	
	RRC (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 001 110	
	RR r		↕	↕	0	P	[illegible]	↕	2	[illegible]	11 001 011 00 011 r	Ⓔ
	RR (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 011 110	
	RR (IX+d)	(CY) → 7 → 0 → CY r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 011 110	
	RR (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 011 110	
	SLA r		↕	↕	0	P	[illegible]	↕	2	8	11 001 011 00 100 r	Ⓔ
	SLA (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 100 110	
	SLA (IX+d)	CY ← 7 ← 0 ← 0 r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 100 110	
	SLA (IY+d)		↕	↕	0	P	0	↕	[illegible]	23	11 111 101 11 001 011 ← d → 00 100 110	

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210
ローテート・シフト命令	SRA r		↕	↕	0	P	0	↕	2	8	11 001 011 00 101 r Ⓔ
	SRA (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 101 110
	SRA (IX+d)	[7→0]→[CY] r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 101 110
	SRA (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 101 110
	SRL r		↕	↕	0	P	0	↕	2	8	11 001 011 00 111 r Ⓔ
	SRL (HL)		↕	↕	0	P	0	↕	2	15	11 001 011 00 111 110
	SRL (IX+d)	0→[7→0]→[CY] r, (HL), (IX+d), (IY+d)	↕	↕	0	P	0	↕	4	23	11 011 101 11 001 011 ← d → 00 111 110
	SRL (IY+d)		↕	↕	0	P	0	↕	4	23	11 111 101 11 001 011 ← d → 00 111 110
	RLD	A [7−4|3−0] [7−4|3−0](HL)	↕	↕	0	P	0	·	2	18	11 101 101 01 101 111
	RRD	A [7−4|3−0] [7−4|3−0](HL)	↕	↕	0	P	0	·	2	18	11 101 101 01 100 111
ビット操作命令	BIT b, r	$Z \leftarrow \overline{r_b}$	×	↕	1	×	0	·	2	8	11 001 011 01 b r ⒺⒻ
	BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	×	↕	1	×	0	·	2	12	11 001 011 01 b 110 Ⓕ
	BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	×	↕	1	×	0	·	4	20	11 011 101 11 001 011 ← d → 01 b 110 Ⓕ
	BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	×	↕	1	×	0	·	4	20	11 111 101 11 001 011 ← d → 01 b 110 Ⓕ
	SET b, r	$r_b \leftarrow 1$	·	·	·	·	·	·	2	8	11 001 011 11 b r ⒺⒻ
	SET b, (HL)	$(HL)_b \leftarrow 1$	·	·	·	·	·	·	2	15	11 001 011 11 b 110 Ⓕ
	SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	·	·	·	·	·	·	4	23	11 011 101 11 001 011 ← d → 11 b 110 Ⓕ
	SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	·	·	·	·	·	·	4	23	11 111 101 11 001 011 ← d → 11 b 110 Ⓕ
	RES b, r	$r_b \leftarrow 0$	·	·	·	·	·	·	2	8	11 001 011 10 b r ⒺⒻ
	RES b, (HL)	$(HL)_b \leftarrow 0$	·	·	·	·	·	·	2	15	11 001 011 10 b 110 Ⓕ
	RES b, (IX+d)	$(IX+d)_b \leftarrow 0$	·	·	·	·	·	·	4	23	11 011 101 11 001 011 ← d → 10 b 110 Ⓕ
	RES b, (IY+d)	$(IY+d)_b \leftarrow 0$	·	·	·	·	·	·	4	23	11 111 101 11 001 011 ← d → 10 b 110 Ⓕ

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	OPコード 76 543 210	
ジャンプ・コール・リターン命令	JP nn	PC←nn	·	·	·	·	·	·	3	10	11 000 011 ← n → ← n →	
	JP cc, nn	If cc is true PC←nn Otherwise continue	·	·	·	·	·	·	3	10	11 cc 010 ← n → ← n →	Ⓒ
	JR e	PC←PC+e	·	·	·	·	·	·	2	12	00 011 000 ← e−2 →	
	JR C, e	If C=0 continue [illegible] C=1 PC←PC+e	·	·	·	·	·	·	2	7 if C=0 12 if C=1	00 111 000 ← e−2 →	
	JR NC, e	If C=1 continue If C=0 PC←PC+e	·	·	·	·	·	·	2	7 if C=1 12 if C=0	00 110 000 ← e−2 →	
	JR Z, e	If Z=0 continue If Z=1 PC←PC+e	·	·	·	·	·	·	2	7 if Z=0 12 if Z=1	00 101 000 ← e−2 →	
	JR NZ, e	If Z=1 continue If Z=0 PC←PC+e	·	·	·	·	·	·	2	7 if Z=1 12 if Z=0	00 100 000 ← e−2 →	
	JP (HL)	PC←HL	·	·	·	·	·	·	1	4	11 101 001	
	JP (IX)	PC←IX	·	·	·	·	·	·	2	8	11 011 101 11 101 001	
	JP (IY)	PC←IY	·	·	·	·	·	·	2	8	11 111 101 11 101 001	
	DJNZ e	B←B−1 if [illegible]=0 continue if B≠[illegible] PC←PC+e	·	·	·	·	·	·	2	8 if B=0 13 if B≠0	[illegible] 010 000 ← e−2 →	
	CALL nn	$(SP-1)←PC_H$ $(SP-2)←PC_L$ PC←nn	·	·	·	·	·	·	3	17	11 001 101 ← n → ← n →	
	CALL cc, nn	If cc is false continue otherwise same [illegible] CALL nn	·	·	·	·	·	·	3	10 if cc is false 17 if cc is true	11 cc 100 [illegible] n → ← n →	Ⓒ
	RET	$PC_L←(SP)$ $PC_H←(SP+1)$	·	·	·	·	·	·	1	10	11 001 001	
	RET cc	If cc is false continue otherwise same as RET	·	·	·	·	·	·	1	5 if cc is false 11 if cc is true	11 cc [illegible]	Ⓒ
	RETI	Return from interrupt	·	·	·	·	·	·	2	14	11 101 101 01 001 101	
	RETN	Return from [illegible] maskable interrupt	·	·	·	·	·	·	2	14	11 101 101 01 000 101	
	RST p	$(SP-1)←PC_H$ $(SP-2)←PC_L$ $PC_H←0$ $PC_L←p$	·	·	·	·	·	·	1	11	11 t 111	Ⓗ
入出力命令	IN A, n	A←(n) $A_{0\sim7}←n$ $A_{8\sim15}←A$	·	·	·	·	·	·	2	11	11 011 011 ← n →	
	IN r, (C)	r←(C) if r=110 only the flags will be affected $A_{0\sim7}←C$ $A_{8\sim15}←B$	↕	↕	↕	P	0	·	[illegible]	12	11 101 101 01 r 000	Ⓔ
	INI	(HL)←(C) B←B−1 HL←HL+1 $A_{0\sim7}←C$ $A_{8\sim15}←B$	×	③ ↕	×	×	1	·	2	16	11 101 101 10 100 010	
	INIR	(HL)←(C) B←B−1 HL←HL+1 until B=0 $A_{0\sim7}←C$ $A_{8\sim15}←B$	×	1	×	×	1	·	2	21 if B≠0 16 if B=0	11 101 101 10 110 010	
	IND	(HL)←(C) B←B−1 HL←HL−1 $A_{0\sim7}←C$ $A_{8\sim15}←B$	×	③ ↕	×	×	1	·	2	16	11 101 101 10 101 010	

命令群	ニーモニック	オペレーション	S	Z	H	P/V	N	C	バイト	ステート	76	543	210
入出力命令	INDR	(HL)←(C) B←B−1 HL←HL−1 until [illegible]=[illegible] $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	×	1	×	×	1	・	2	21 if B≠0 16 if B=0	11 10	101 111	101 [illegible]
	OUT n, A	(n)←A $A_{0\sim7}\leftarrow n$ $A_{8\sim15}\leftarrow A$	・	・	・	・	・	・	2	11	11 ←	010 n	011 →
	OUT (C), r	(C)←r $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	・	・	・	・	・	・	2	12	11 01	101 r	101 001 Ⓔ
	OUTI	(C)←(HL) B←B−1 HL←HL+1 $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	×	③ ↕	×	×	1	・	2	16	11 [illegible]	101 100	101 011
	OTIR	(C)←(HL) B←B−1 HL←HL+1 until B=0 $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	[illegible]	1	×	×	1	・	[illegible]	21 if B≠0 16 if B=0	11 [illegible]	101 110	101 011
	OUTD	(C)←(HL) B←B−1 HL←HL−1 $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	×	③ ↕	×	×	1	・	2	16	11 10	101 101	101 011
	OTDR	(C)←(HL) B←B−1 HL←HL−1 until B=0 $A_{0\sim7}\leftarrow C$ $A_{8\sim15}\leftarrow B$	×	1	×	×	1	・	2	21 if B≠0 16 if B=0	11 10	101 111	101 011

Ⓐ		Ⓑ		Ⓒ		Ⓓ		Ⓔ		Ⓕ		Ⓖ				[illegible]	
Reg	ss dd	Reg	[illegible]	Reg	pp	Reg	rr	Reg	r,r'	Bit	b	cc		Condition	Flag	P	t
BC	00	BC	00	BC	00	BC	00	B	000	0	000	[illegible]	NZ	Non Zero	Z	00H	[illegible]
DE	[illegible]	DE	01	DE	[illegible]	DE	01	C	001	1	001	001	Z	Zero	Z	08H	001
HL	10	HL	10	IX	10	IY	10	D	010	[illegible]	010	010	NC	Non Carry	C	10H	010
SP	11	AF	11	SP	11	SP	11	E	011	3	011	011	C	Carry	C	18H	011
								H	100	[illegible]	100	100	PO	Parity Odd	P/V	20H	100
								L	101	5	101	101	PE	Parity Even	P/V	28H	101
								A	111	6	110	110	P	Sign Positive	S	30H	110
										7	111	111	M	Sign Negative	[illegible]	38H	111

d, n ：8ビット・イミーディエト・データ
e　：相対アドレシングの変位置
e−2：eの実効変位置

フラグ

・：影[illegible]受けない　↕：演[illegible]結果に従った影響を受ける　① BC−1=0 ならば P/V=0, その他 P/V=1
0：リセット　P："1" 偶数パリティ, "0" 奇数パリティ　② A=(HL) ならば Z=1, その他 Z=0
1：セット　V："[illegible]"オーバフロー有り, "0"オーバフロー無し　③ B−1=0 ならば Z=1, その他 Z=0
×：不定　IFF：P/Vフラグ←(IFF)

Z80 ニーモニック←→マシン語対照表

8ビット・ロード

×	I	R	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX+d)	(IY+d)	(nn)	n
LD A, ×	ED 57	ED 5F	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3A n n	3E n
LD B, ×			47	40	41	42	43	44	45	46			DD 46 d	FD 46 d		06 n
LD C, ×			4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d		0E n
LD D, ×			57	50	51	52	53	54	55	56			DD 56 d	FD 56 d		16 n
LD E, ×			5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d		1E n
LD H, ×			67	60	61	62	63	64	65	66			DD 66 d	FD 66 d		26 n
LD L, ×			6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d		2E n
LD (HL), ×			77	70	71	72	73	74	75							36 n
LD (BC), ×			02													
LD (DE), ×			12													
LD (IX+d), ×			DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d							DD 36 d n
LD (IY+d), ×			FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d							FD 36 d n
LD (nn), ×			32 n n													
LD I, ×			ED 47													
LD R, ×			ED 4F													

16ビット・ロード

×	AF	BC	DE	HL	SP	IX	IY	nn	(nn)
LD AF, ×									
LD BC, ×								01 n n	ED 4B n n
LD DE, ×								11 n n	ED 5B n n
LD HL, ×								21 n n	2A n n
LD SP, ×				F9		DD F9	FD F9	31 n n	ED 7B n n
LD IX, ×								DD 21 n n	DD 2A n n
LD IY, ×								FD 21 n n	FD 2A n n
LD (nn), ×		ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n		
PUSH ×	F5	C5	D5	E5		DD E5	FD E5		
POP ×	F1	C1	D1	E1		DD E1	FD E1		

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

ブロック・サーチ

CPI	ED A1
CPIR	ED B1
CPD	ED A9
CPDR	ED B9

ローテート，シフト

×	A	B	C	D	E	H	L	(HL)	(IX +d)	(IY +d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	C[illegible] 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

	A
RLCA	07
RRCA	0F
RLA	17
RRA	1F

ジャンプ，コール，リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR ×, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL ×, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET ×	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

ビット操作

×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

入　力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

出　力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

PC-8801＋mkⅡ
マシン語活用入門

発行――1986年 5 月15日
著者――塚本浩二
発行者――田村正隆
発行所――株式会社ナツメ社
郵便番号101
東京都千代田区神田神保町1-52
電話（03）291-1257
振替 東京3-58661
（落丁・乱丁本はお取り替えします）

定価――1200円

PC-8801+mkII

マシン語活用入門

ナツメ社
定価=1200円

[目次より]

1・1——マシン語とは何か——8
1・2——なぜ、マシン語なのか——12

2・1——マシン語は難しくない——20
2・2——ニーモニックは人間的だ——22
2・3——マシン語マスターへの近道——27
2・4——内部ルーチンを使え！——30

3・1——この章で学ぶこと——40
3・2——マシン語の基礎知識——41
3・3——レジスタとは——49
3・4——フラグとは何か——52
3・5——LD命令——55
3・6——演算命令——59
3・7——JP命令とCP命令——64
3・8——CALL命令とRET命令——70

4・1——本章を読む前に——78
4・2——アセンブラのための命令——79
4・3——内部ルーチンの活用！——85
4・4——マシン語からBASICへ——92
4・5——マシン語からモニタへ——97
4・6——WIDTHをマシン語で——101
4・7——画面設定内部サブルーチンあれこれ——113
4・8——PRINTするにはどうするか——123
4・9——1文字表示内部ルーチン——128
4・10——マシン語で文字列をPRINT！——149

5・0——おわりに——160

ISBN4-8163-0367-7 C2054 ¥1200E